НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Забезпечення контроля версій клієнтських даних в CRM системах

Виконав (-ла): студент (-ка) 2 курсу, групи ФБ-71мп
(шифр групи)

Артеменко Валерій Тарасович
(прізвище, ім’я, по батькові)

Науковий керівник к.т.н., доц. Стьопочкіна Ірина Валеріївна _ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент ФІОТ Жданова О.Г. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬННЙ ТЕХНІЧННЙ УНІВЕРСНТЕТ УКРАЇНН
«КНЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗНКО-ТЕХНІЧННЙ ІНСТНТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 125 Кібербезпека («Системи і технології кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Артеменку Валерію Тарасовичу

1. Тема дисертації: Забезпечення контроля версій клієнтських даних в CRM системах
науковий керівник дисертації к.т.н., доц. Стьопочкіна Ірина Валеріївна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15» листопада 2018 р. № 4171-с

2. Термін подання студентом дисертації 12.12.2018 р.

3. Об'єкт дослідження _____
_____

4. Вихідні дані _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____
_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____
_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Робота обсягом 104 сторінок містить 16 ілюстрацій, 22 таблиці та 24 літературних посилань.

Метою даної кваліфікаційної роботи є дослідження механізмів забезпечення цілісності, доступності та спостережності версій клієнтських даних в CRM системах; розробка рішень з інтеграції системи керування версіями в CRM систему.

Об'єктом дослідження є CRM системи, і їх конкретний приклад - SalesForce система.

Предметом дослідження є програмні продукт збереження версій файлу, виявлення зміненої інформації.

Результати роботи викладені у вигляді програмного застосунку для SalesForce системи для збереження контролю версій фалів в середині системи.

Результати роботи можуть застосовуватися представниками користувачів хмарного продукту SalesForce для хронології поведінки файлу, а саме для вирішення ряду задач, які передбачають відновлення перетертих, знищених чи Випадково видалених даних.

СИСТЕМА КОНТРОЛЮ ВЕРСІЙ, ІНТЕРФЕЙС ПРОГРАМУВАННЯ ЗАСТОСУНКІВ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПЕРЕДАЧА СТАНУ УЯВЛЕННЯ, БАЗА ДАНИХ.

РЕФЕРАТ

Работа обмелом 104 страниц содержит 16 иллюстраций, 22 таблицы и 24 литературных ссылок.

Целью данной квалификационной работы является исследование механизмов обеспечения целостности, доступности и наблюдаемости версий клиентских данных в CRM системах; разработка решений по интеграции системы управления версиями в CRM систему.

Объектом исследования является CRM системы, и их конкретный пример - SalesForce система.

Предметом исследования является программные продукт сохранения версий файла, выявление измененной информации.

Результаты работы изложены в виде программного приложения для SalesForce системы для сохранения контроля версий фалов внутри системы.

Результаты работы могут применяться представителями пользователей облачного продукта SalesForce для хронологии аовединкы файла, а именно для решения ряда задач, которые предусматривают восстановление перетертых, уничтоженных или случаи удаленных данных.

СИСТЕМА КОНТРОЛЯ ВЕРСИЙ, ИНТЕРФЕЙСА ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ПЕРЕДАЧА СОСТОЯНИЯ ПРЕДСТАВЛЕНИЯ, БАЗА ДАННЫХ.

ABSTRACT

The work includes 104 pages, 16 figures, 22 tables, 24 literary references.

The purpose of this qualification work is to study the mechanisms for ensuring the integrity, availability and observability of versions of client data in CRM systems; development of solutions for integrating a version control system into a CRM system.

The object of the study is the CMS system, and their specific example is the SalesForce System.

The subject of the study is a software product for saving versions of the file, detecting altered information.

The results of the work are presented as a software application for the SalesForce system to keep control of the file versions in the middle of the system.

The results of the work can be used by representatives of the users of the cloud SalesForce product for the history of the file, in particular for solving a number of tasks that involve the restoration of overturned, destroyed, or

Accidental deleted data.

VERSION CONTROL SYSTEM, INTERFACE APPLICATION PROGRAM, SOFTWARE, TRANSFER STATUS, DATA BASIS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – Програмне забезпечення

CRM – Customer Relationship Management;

CSV – Система контролю версій;

SF – SalesForce;

REST - Передача стану уявлення;

API - Інтерфейс програмування застосунків;

ВСТУП

Кожен день користувач робить свій вибір кому довіряти свої дані та на яких сервісах їх зберігати, так як такі платформи дають гарантію безпечного збереження даних користувача, використовуючи апаратні та програмні засоби захисту даних. Виникає питання у необхідності автоматизації різних процесів пов'язаних з обробкою даних, де найкращим інструментом є CRM системи.

Користувачі CRM систем за часту працюють над окремими функціональними модулями, але бувають випадки коли різні модулі можуть мати доступ та використовувати спільні файли, при перезаписі яких попередні функціональні модулі можуть втратити свої властивості. Вирішення таких ситуацій може запропонувати система керування версіями, що надає можливість керування версіями одиниці інформації, вихідного коду програми, скрипту, веб-сторінки, веб-сайту, текстового документу.

Актуальність роботи Актуальність проблеми полягає в тому, що користувачі CRM систем працюють над розробкою спільного функціоналу в CRM системах, при чому виникає проблема контролю версій файлу, так як користувачі, які мають доступ до спільного файлу, можуть перезаписувати його в різний момент часу, що в свою чергу впливає на поведінку певного функціоналу.

Метою роботи є дослідження механізмів забезпечення цілісності, доступності та спостережності версій клієнтських даних в CRM системах; розробка рішень з інтеграції системи керування версіями в CRM систему.

Задачі дослідження:

- Визначити основні принципи роботи з клієнтськими даними в CRM системах, існуючі механізми забезпечення цілісності, конфіденційності, доступності та спостережності даних.

- Дослідити принципи роботи систем контролю версій, визначити основні переваги та недоліки існуючих рішень.

- Визначити основні підходи до інтеграції системи контролю версій в склад CRM системи;

- Запропонувати елементи системи контролю версій в умовах, типових для даних, що обробляються CRM системою. Зокрема, для вимог цілісності та конфіденційності та доступності версій: обрати метод стиснення даних, шифрування та ЕЦП;

- Запропонувати архітектурні рішення для реалізації контролю версій в складі CRM системи. Розробити відповідний програмний продукт для обраної CRM системи.

- Імплементувати систему контроля версій до CRM системи на прикладі SalesForce CRM системи.

Об'єктом дослідження є CRM системи, і їх конкретний приклад - SalesForce System.

Предметом дослідження є системи контролю версій даних .

Практичне значення роботи полягає у створенні власної системи контролю версій з можливістю впровадження для подальшого використання, імплементація розробленої системи в SalesForce середовищі. Одержані в ході роботи результати: властивості та основні функції розробленої системи, механізм роботи, алгоритм стиснення, алгоритм збереження даних, основні можливості системи можуть бути використані розробниками в сфері CRM систем.

1 ВИКОРИСТАННЯ ФАЙЛОВИХ ХМАРНИХ СИСТЕМАХ ТА СИСТЕМУ КОНТРОЛЯ ВЕРСІЙ В ЗАДАЧАХ КІБЕРБЕЗПЕКИ

1.1 Огляд літературних джерел

Для вирішення поставленого завдання опрацьовано літературні джерела, що розкривають принципи роботи CRM систем її можливості як її використання, основні принципи.

Причина виникнення технології CRM є досить простою. З кожним роком все складніше ставало ефективно відстежувати історію роботи з клієнтами. Чим більше компанія, тим більше у неї інформації про клієнтів і тим складніше її структурувати і обробляти. У 90-ті роки відбулася істотна зміна в системі взаємовідносин з клієнтами, і саме в цей час вперше був представлений цей термін - CRM.

Першим джерелом за допомого якого я правильно підійшов для розгортання CRM системи була книга Алексія Рязанцева “Як впровадити CRM - систему за 50 днів“. В даній книзі я ознайомився з причиною появи даних систем. як правильно розгорнути та с чого потрібно почати початківцю з роботою з CRM системами, вирішити для яких цілей буде використовуватися система і як налагодити систему під індивідуального користувача

Більш детальну структуру CRM систем описано о посібнику для розробників “Microsoft Dynamics CRM», де чітко описується бізнес логіка, організаційна структура даних, безпечність, опис бази даних та як вона влаштована, основні бізнес процеси, та можливі варіанти розширення можливостей бізнес процесів.

Наступне цікаве джерело з яким мені вдалося вдало опрацювати був офіційний Git – хостинг, який я використовував щоб зрозуміти як працюють сучасні системи контролю версій, основні їх принципи, методі стиснення та шифрування даними що стало основним поштовхом для розробки власного інструменту контролі версій в CRM системі.

1.2 Огляд роботи сучасних хмарних системи, управління даними.

На сьогоднішній день роль інформації, а також процеси її обробки, передачі, накопичення та ін. Незмірно зросли, впливаючи на науково-технічний потенціал країни, рівень економічного розвитку, спосіб життя і діяльності людини відповідно до процесами, що відбуваються інформатизації суспільства як «глобального соціального процесу, особливість якого полягає в тому, що домінуючим видом діяльності в сфері суспільного виробництва є збір, накопичення, обробка, зберігання, передача, використання , здійснювані на основі сучасних засобів мікропроцесорної та обчислювальної техніки, а також різноманітних засобів інформаційної взаємодії та обміну »[1]. Відзначаючи ключові тенденції IT-галузі, такі як великі дані, «інтернет речей», цифрове виробництво, мобільність, кібербезпека і ін. Виділимо напрямок хмарних технологій, під якими будемо розуміти галузь обчислювальних технологій, що забезпечують на вимогу користувача віддалений доступ до цілого набору обчислювальних ресурсів (додатків, сервісів, сховищ даних), розташованих на серверах в мережі Інтернет. При цьому доступ до ресурсів надається користувачам за допомогою хмарних сервісів - безкоштовних або умовно безкоштовних хмарних додатків, програмні та апаратні вимоги яких не припускають наявності у клієнтів високопродуктивних і ресурс ємних комп'ютерів.

Сама концепція хмарних технологій не нова: ще в 1960 р подібні ідеї були озвучені поруч вчених (Джозеф Ліклайдер, Джоном Маккарті і ін.) З метою надання користувачам мережі різного роду послуг. Однак, у зв'язку з недостатнім рівень розвитку інструментальних засобів в сфері IT, реалізація даної концепції була неможлива аж до 90-х років. Зміни, що відбулися в той час зміни в області телекомунікаційних технологій, такі як, перш за все, збільшення пропускної здатності мережі, і, як наслідок, збільшення швидкості обміну даними, зниження вартості Інтернет трафіку і ін. Уможливили і доступною

область хмарних технологій. Слід зазначити, що виникнення хмарних технологій зобов'язана не тільки збільшення пропускної здатності мережі, а й об'єднання в єдиному технологічному вирішенні безлічі перспективних технологій, таких як [2]:

- технології віртуалізації, під якими розуміється оренда віртуальних серверів на чужому обладнанні, в наслідок чого знижується вартість підтримки дорогої IT-інфраструктури;

- сервіс-орієнтована архітектура (Service-OrientedArchitecture, SOA), що забезпечує гнучкий процес взаємодії викладача та студентів при побудові віртуальних освітніх середовищ;

- grid-обчислення, що представляють географічно розподілену інфраструктуру, доступ до ресурсів якої можливий з будь-якої точки, незалежно від місця їхнього економічного становища, що відкриває широкі можливості створення освітніх середовищ;

- SaaS-технології, орієнтовані на застосування інтернет-технологій, що надають не ПЗ, а реалізацію необхідних функцій в розподілених моделях навчання;

- системи розробки (developmentframework), що дозволяють створювати власні програми, що працюють на стороні провайдера і які доставляються користувачам через Інтернет;

- ПЗ з відкритим кодом, які є ідеальним середовищем навчання найсучаснішим комп'ютерним технологіям;

- utilitycomputing - надання віртуальних серверів після проведення оплати за їх використання; - сервіси Web 2.0, що працюють за принципом залучення користувачів до наповнення і редагування контенту.

Так чи інакше, хмарні сервіси присутні в житті користувачів Інтернету вже давно: хостинги, хмарні сховища, віртуальні дошки, онлайн засоби створення, редагування і публікації презентацій, графіки, колажі, анімації, відео, офісні технології по роботі з документами, конвертори, календарі, органайзери, планувальники, інформери і закладки, технології скрайбінга і

візуалізації, скрінкасти, стрічки часу, вебінари, онлайн-конференції, карти знань і географічні карти, онлайн-бібліотеки, блоги, соціальні мережі, тести, додаток в браузері, віртуальна реальність, LMS-системи, конструктори уроків та ін. - неповний перелік сучасних сервісів, орієнтованих на віддалений доступ до контенту і спільну роботу користувачів. Наприклад, якщо раніше для використання можливостей електронної пошти було необхідна наявність відповідного програмного забезпечення (ПЗ), встановленого на локальному комп'ютері користувача, тепер стає можливим використання спеціалізованого веб-сервісу (gmail.com, mail.ru, yandex.ru і ін.), що дозволяє отримати доступ до електронної пошти з будь-якого пристрою, що має доступ до мережі Інтернет.

Відповідно до зростаючим поширенням і популярністю хмарних технологій, а також їх перевагами в порівнянні з фізичними серверами (доступність, мобільність, економічність, гнучкість, висока технологічність, надійність і ін.), На особливу увагу заслуговують питання безпеки подібних рішень, конфіденційності збережених даних, а також протистояння різним видам загроз (мережеві атаки, уразливості в додатках операційних систем, шкідливе програмне забезпечення та ін.). На вирішення цього питання орієнтовані безліч фахівців різних рівнів і областей, відповідно до чого на даний момент існують комплексні заходи та ефективні інструменти захисту даних.

Слід також зазначити, що в контексті хмарної парадигми, хмарні технології існують нерозривно з користувачем, що визначає необхідність розгляду взаємодії користувача і хмари як єдиної системи. Відповідно до чого, відповідальність за безпеку і конфіденційність даних є областю діяльності не тільки провайдера-постачальника хмарних послуг, але і самого користувача. Безпека повинна забезпечуватися по всьому ланцюжку, починаючи від провайдера хмарного сервісу і до споживача послуг, включаючи зв'язують засоби їх комунікації. Головними завданнями провайдера є забезпечення безпеки як фізичної, так і програмної частини, а також забезпечення цілісності інформації та недоторканності даних від посягань третіх осіб. У свою чергу,

користувач хмарних послуг також повинен забезпечити персональні дані від можливості отримання прав доступу третіми особами

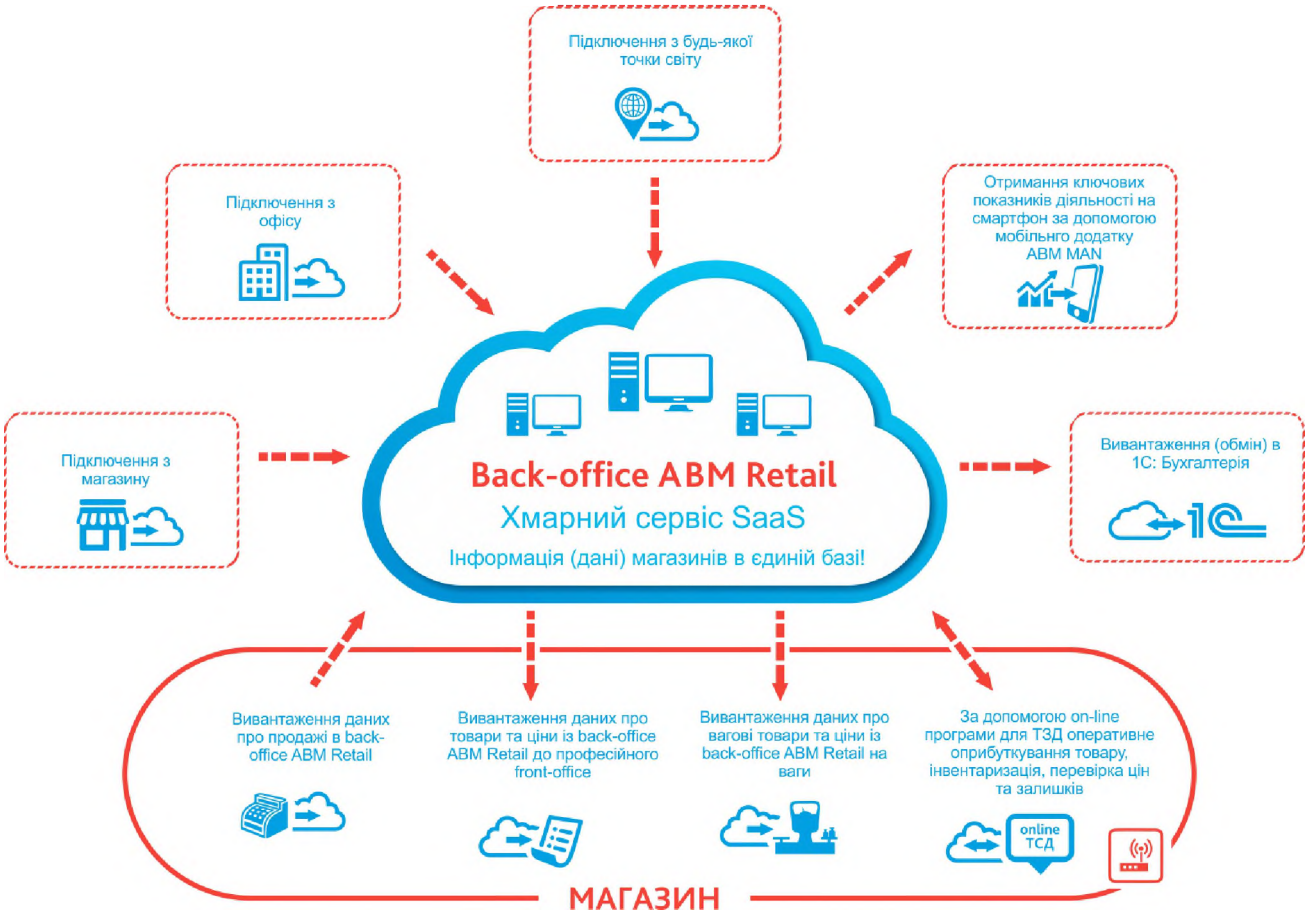

Рисунок — 1.1 Архітектура організації даних в «хмарній системі»

В даний час «в теорії і практиці інформаційної безпеки вже починають викристалізовуватися два напрямки, які можна визначити (можливо, ще в деякій мірі умовно) як інформаційно-психологічна безпека і захист інформації» [2]. Інформаційно-технічний прогрес торкнувся всі сфери діяльності, приніс багато позитивних плодів. І, звичайно, приніс деякий фронт ризиків, пов'язаний із захистом інформації. Ні для кого не секрет, що в XXI столітті невід'ємну частину життя суспільства складають "Хмарні технології".

У даному розділі буде мова про захист інформації при використанні самих інноваційних технологій, а саме хмарних. Безпека інформації здійснюється за умови забезпечення її конфіденційності, доступності та цілісності. Організаційно-технічними методами забезпечення інформаційної безпеки є: створення і вдосконалення системи забезпечення інформаційної

безпеки, розробка, використання та вдосконалення засобів захисту інформації, створення систем і засобів запобігання несанкціонованому доступу до оброблюваної інформації, а також виявлення технічних пристроїв і програм, які становлять небезпеку для нормального функціонування IT-систем.

Постійно зростаючі витрати на створення і експлуатацію інформаційних систем, істотне зростання збитку від інформаційних ризиків змушують керівників шукати нові шляхи підвищення ефективності інформаційної сфери підприємств і організацій. Одним із сучасних напрямків підвищення ефективності використання інформаційних систем є перехід до Cloud computing. В Україні термін "Cloud computing" в результаті прямого перекладу отримав трактування - "Хмарні обчислення". Слід зауважити, що дослівний переклад виразу Cloud computing як "Хмарні обчислення" в повному обсязі відображає сутність сучасних процесів віддаленого обслуговування користувачів інформаційних систем. Cloud computing, крім віддаленого виконання програм, передбачає весь комплекс інформаційних послуг, включаючи зберігання, пошук і передачу інформації, забезпечення її безпеки та багато іншого. Тому доречніше вживати термін "Хмарні технології". Хмарні технології "(Cloud computing) зараз перебувають на хвилі популярності. Економічність, легкість розгортання, розрахована на багато користувачів архітектура - все це сприяє швидкому їх поширенню, а також передбачає весь комплекс інформаційних послуг, включаючи зберігання, пошук і передачу інформації, забезпечення її безпеки та багато іншого . [1]

"Хмарні технології" - це підхід до розміщення, надання і споживання додатків і комп'ютерних ресурсів, при якому додатки і ресурси стають доступні через Інтернет у вигляді сервісів, споживаних на платформах і пристроях. [1]

Основними характеристиками хмарних обчислень є:

- **еластичність** (дозволяє швидко наростити потужність інфраструктури без впровадження інвестицій в обладнання та програмне забезпечення);

- **самообслуговування** (дозволяє споживачам запросити й одержати необхідні ресурси за лічені хвилини).

- **виділена архітектура**. Кожна група користувачів використовує свою копію програми (сервісу). Кожна копія орієнтована тільки на свою групу користувачів, таким чином, кожна копія містить персональне розширення, в якому зберігаються призначені для користувача настройки. Вся неефективність даної моделі стане наочна при збільшенні числа користувачів. Також стає неможливим підвищення ефективності за рахунок зростання масштабів, оскільки фактично користувачі користуються різними екземплярами програми.

- **налаштована архітектура**. В даному випадку додаток, або сервіс налаштовується для кожного конкретного користувача через конфігурацію. Тобто кожен користувач працює зі своєю копією додатка, не дивлячись на те, що копії ідентичні. Обчислювальні потужності не розділяються між різними екземплярами додатки, що також не дозволяє домогтися підвищення ефективності за рахунок збільшення масштабів .. Розділення ж самого додатка може бути як віртуальним, так і фізичним.

- **мультітенантна архітектура**. Налаштування для кожного користувача виконується виключно через конфігурацію, при використанні інструментів для самостійного налаштування. При цьому відсутня можливість горизонтального масштабування, тобто підвищення продуктивності може бути досягнуто тільки через вертикальне масштабування.

- **масштабована архітектура**. Дана модель підтримує мультітенатний підхід, а також можливість горизонтального масштабування. При цьому нові екземпляри додатків додаються до загального пулу. Загальне навантаження розподіляється по всій інфраструктурі. Архітектура налаштовується через конфігурацію.

Моделі організації мультітенантного сховища даних

1. Окремі бази даних. Кожна група користувачів має власну базу даних, з власної схемою даних. Може бути ефективною при невеликому числі користувачів в розрахунку на базу даних. Крім того, даний варіант кращий, в разі пред'являються користувачами строгих вимог до ізоляції і безпеки даних.

2. Спільно використовувані бази даних з різними схемами. Всі користувачі працюють з однією БД, але мають різні набори зумовлених полів. Найбільш корисний даний варіант у разі, якщо зберігання даних різних користувачів в одній таблиці допустимо, а також заздалегідь відомо які поля будуть потрібні. Може привести до ситуації розрядженого наповнення таблиць.

3. Спільно використовувані бази даних зі спільною схемою. Для зберігання розширень даних використовуються спеціальні підходи і техніки, користувачі працюють з однією і тією ж базою даних. При цьому, можна запропонувати користувачам практично необмежену кількість полів, однак, виникають проблеми при організації індексації та пошуку інформації. Даний варіант застосуємо, якщо зберігання даних різних груп користувачів в одних і тих же таблицях задовольняє вимогам безпеки та ізоляції даних. Найбільш ефективними способами забезпечення безпеки "Хмарних технологій" є:

Збереження даних. Шифрування

Шифрування - один з найефективніших способів захисту даних. Провайдер, що надає доступ до даних, повинен шифрувати інформацію клієнта, що зберігається в ЦОД (Центр обробки даних - сукупність серверів, розміщених на одному майданчику з метою підвищення ефективності і захищеності), а також у разі відсутності необхідності, безповоротно видаляти. [1] При шифруванні даних завжди виникає питання про ключі. Їх зберігання на хмарному сервері недоцільно, оскільки кожен, хто має доступ до хмарних серверів або шаблонами, міг би отримати доступ до ключа, а значить, і до розшифрованих даними. Набір пароля при запуску системи, як це прийнято в локальних рішеннях для шифрування даних, утруднений у зв'язку з відсутністю

справжньої консолі, однак ідея непогана. Фізичний введення ключа замінюється запитом, який хмарний сервер відправляє зовнішнього джерела - сервера управління ключами (Key Management Server, KMS). [3]

Вирішальним фактором для забезпечення безпеки такого рішення є роздільна експлуатація хмарного сервера і сервера управління ключами (див. Малюнок 1): якщо обидва розміщені у (одного і того ж) провайдера хмарних сервісів, то вся інформація знову виявляється зібраної в одному місці. Хорошою альтернативою є установка сервера KMS в локальному ЦОД або в якості зовнішньої послуги в іншого сервіс-провайдера. [4]

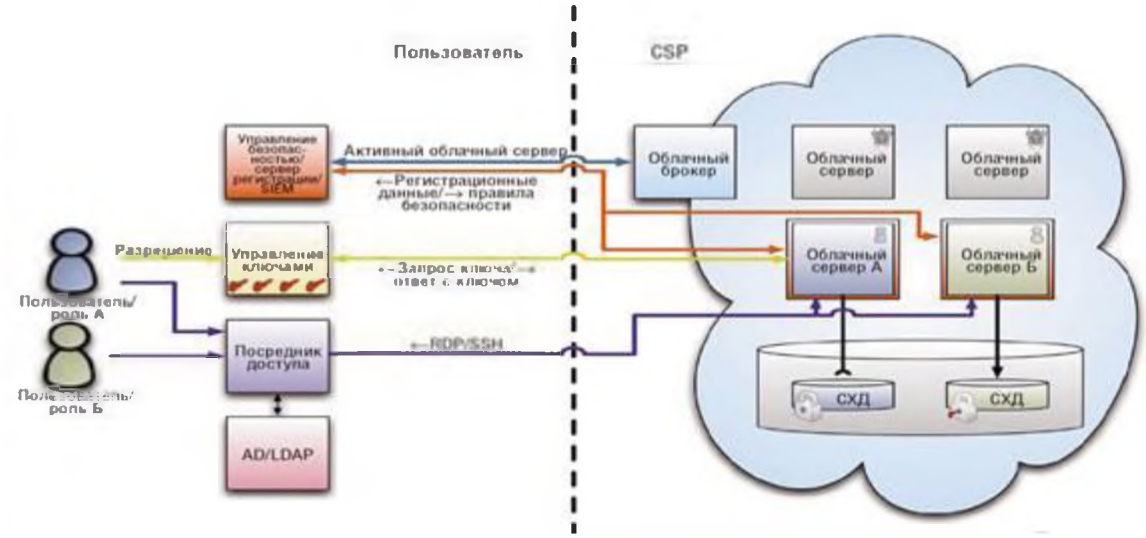

Рисунок — 1.2 Схема взаємодії користувача, сервера управління ключами і хмарного сервера

Захист даних при передачі

Для безпечної обробки даних обов'язковою умовою є їх шифруюча передача. З метою захисту даних в публічному хмарі використовується тунель віртуальної приватної мережі (VPN), що зв'язує клієнта і сервер для отримання публічних хмарних послуг. VPN-тунель сприяє безпечним з'єднанням і дозволяє використовувати єдине ім'я і пароль для доступу до різних хмарним ресурсів. Як засіб передачі даних в публічних хмарах VPN - з'єднання використовує загальнодоступні ресурси, такі як Інтернет. Контроль здійснюється шляхом режимах доступу з шифруванням за допомогою двох

ключів на базі протоколу Secure Sockets Layer (SSL). Більшість протоколів SSL і VPN в якості опції підтримують використання цифрових сертифікатів для аутентифікації, за допомогою яких перевіряється ідентифікаційна інформація іншого боку, причому ще до початку передачі даних. Такі цифрові сертифікати можуть зберігатися на віртуальних жорстких дисках в зашифрованому вигляді, і використовуються вони тільки після того, як сервер управління ключами перевірить ідентифікаційну інформацію і цілісність системи. Отже, такий ланцюжок взаємозалежностей дозволить передавати дані тільки тим хмарним серверів, які пройшли попередню перевірку. Зашифровані дані при передачі повинні бути доступні тільки після аутентифікації. Дані не вийде прочитати або зробити зміни в них, навіть у випадку доступу через ненадійні вузли. Такі технології досить відомі, алгоритми і надійні протоколи AES, TLS, IPsec давно використовуються провайдерами.

Автентифікація

Автентифікація - захист паролем. Для забезпечення більш високої надійності, часто вдаються до таких засобів, як токени (електронний ключ для доступу до чого-небудь) і сертифікати. Найбільш простий і досить надійний метод автентифікації - це технологія одноразових паролів (One Time password, OTP). Такі паролі можуть генеруватися або спеціальними програмами, або додатковими пристроями, або сервісами, з пересилкою користувачеві по SMS. Основна відмінність хмарної інфраструктури полягає у великій масштабованості і ширшій географічній розподіленості. На перший план виходить використання для отримання одноразових паролів мобільних гаджетів, які сьогодні є практично у кожного. У найпростішому випадку одноразовий пароль буде створений спеціальним сервером автентифікації і висланий в SMS на мобільний телефон користувача після введення правильного статичного пароля на сторінку доступу до хмарного сервісу. Для прозорої взаємодії провайдера з системою ідентифікації при авторизації, також рекомендується використовувати протокол LDAP (Lightweight Directory Access Protocol) і мову програмування SAML (Security Assertion Markup Language).

Ізоляція користувачів

Використання індивідуальної віртуальної машини і віртуальної мережі. Віртуальні мережі повинні бути розгорнуті із застосуванням таких технологій, як VPN (Virtual Private Network), VLAN (Virtual Local Area Network) і VPLS (Virtual Private LAN Service). Часто провайдери ізолюють дані користувачів один від одного за рахунок зміни коду в єдиній програмному середовищі. Цей підхід має ризики, пов'язані з небезпекою знайти дірку в нестандартному коді, що дозволяє отримати доступ до даних. У разі можливої помилки в коді користувач може отримати доступ до інформації іншого користувача. Останнім часом такі інциденти часто мали місце. [3] "Хмарні технології" представляють собою значний прогрес у сфері розвитку інформаційних технологій та сервісів. Забезпечуючи на вимогу користувача доступ до загальних джерел обчислювальних ресурсів в автономному, динамічно масштабованому і вивіреному режимі, хмарні обчислення пропонують очевидні переваги в швидкості, оперативності та ефективності роботи з інформацією. У даній технології безпеку грає найважливішу роль, цю проблему фахівці приділяють особливу увагу. Але, незважаючи на всі складності в області безпеки, переваги надаються через Інтернет сервісів переважують можливі ризики і "Хмарні технології" будуть широко затребувані на ринку інформаційних технологій.

Таким чином, інформаційна безпека - дуже серйозна проблема всього людства. У наш час ми часто обробляємо, зберігаємо або передаємо дані, які підкоряються регулюючим і нормативним вимогам. Якщо дані потрапляють під дію нормативних або регулюючих обмежень, то вибір розгортання в хмарі (приватному, гібридному або загальнодоступному) залежить від розуміння того, повністю чи постачальник відповідає цим вимогам. В іншому випадку виникає ризик порушення конфіденційних, нормативних та інших правових вимог. Питання забезпечення безпеки інформації істотні, коли мова йде про конфіденційність. На сьогоднішній день найбільш популярні чотири методи захисту інформації в "Хмарних технологіях":

1) Шифрування;

2) Захист даних при передачі;

3) Автентифікація;

4) Ізоляція користувачів.

1.3 Огляд можливих сучасних рішень та продуктів, їхне порівняння.

Розробка програмного забезпечення (ПЗ) досить трудомісткий процес. Будь-яке ПЗ, виключаючи зовсім невеликі програми, необхідно підтримувати. Вихідний код, файли збірки і конфігурацій, а також документацію потрібно відстежувати, тобто вести історію змін. Особливо зараз, коли ринок змушує творців ПЗ розробляти кілька версій продукту, орієнтованих на різні платформи. Підтримка декількох версій ПЗ — непросте завдання. Створюючи і вносячи зміни в «основну» версію продукту, слід також переконатися, що і «побічні» версії продукту оновлені. З плином часу процес відстеження та внесення змін тільки ускладнюється. Для цих цілей використовуються системи контролю версій. Вибір підходящої системи значно полегшить процес підтримки ПЗ та допоможе уникнути безлічі помилок. Мета і завдання даної статті познайомити читача з найбільш популярними системами контролю версій і дати порівняльний огляд на них.

Системи контролю версій

Система контролю версій - це система, яка реєструє всі зміни в файлах, а в подальшому дозволяє повернутися до їх більш раннім версіями і визначити, ким і коли були зроблені конкретні зміни. Існує два типи систем контролю версій: централізовані і розподілені. У централізованих системах контролю версій (CVS [1], Subversion [2], Perforce) є центральний сервер, на якому зберігаються всі дані, і ряд клієнтів, які отримують з нього копії файлів. У таких системах завжди зрозуміло, хто чим займається в проекті, і їх простіше адмініструвати. Розподілені системи контролю версій (Git [3], Mercurial [4], Bazaar) замість традиційної клієнт-серверної моделі використовують

розподілену, особливістю якої є те, що всі зміни зберігаються в локальному сховищі на комп'ютері і при необхідності синхронізуються з іншими.

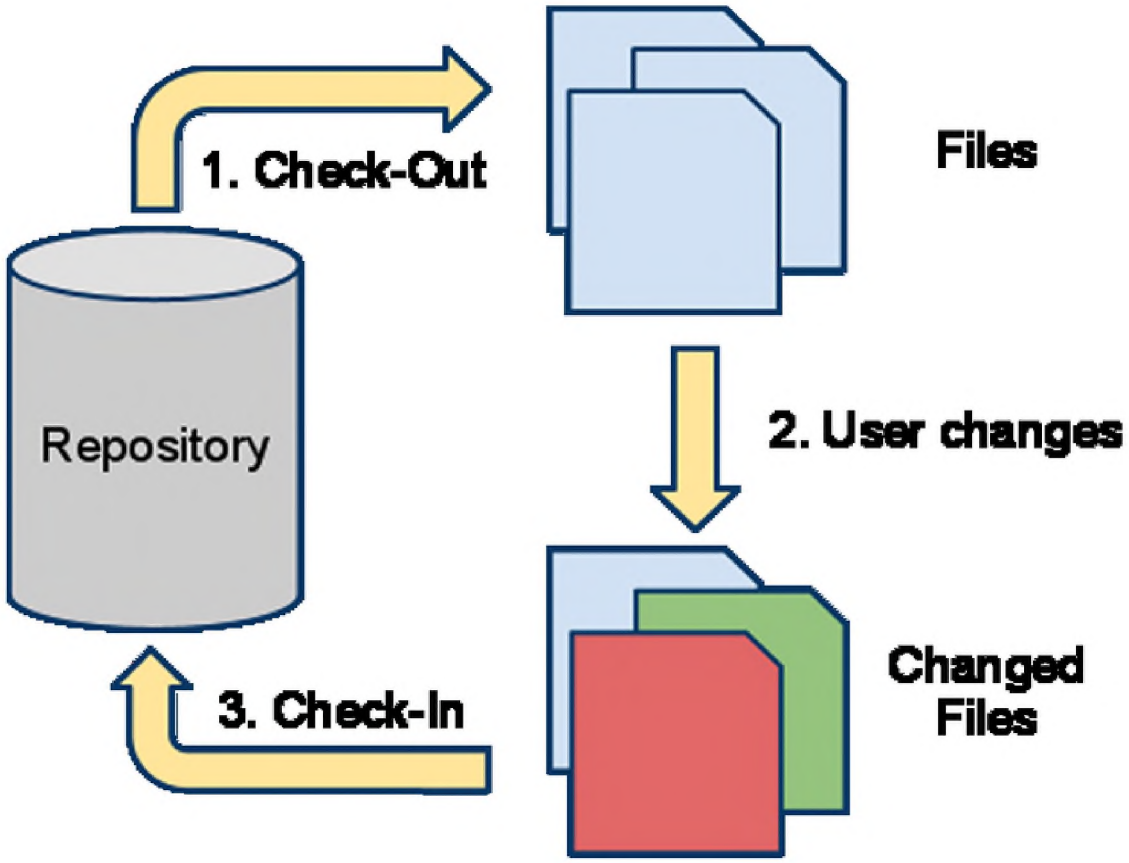

Рисунок — 1.3 Схематичне представлення роботи сучасних рішень

У цьому випадку усувається залежність від центрального сервера, можна вести роботу навіть без підключення до мережі з ним. Існує багато різних систем контролю версій, але в даній статті ми розглянемо тільки чотири найбільш популярних з них [5] [6]. Це Git, Mercurial, Concurrent Versions System (CVS) і Apache Subversion (SVN).

Основні поняття

Сучасні системи контролю версій дозволяють вести розробку в декількох «гілках». Пояснимо це поняття на прикладі. Припустимо, ми розробляємо програму для перегляду графічних зображень. Перед початком розробки ми створили «гілку» master, призначення якої — зберігання

підсумкових версій програми. У першій версії нашого продукту можна переглядати зображення формату PNG. Але ми б хотіли, щоб наша програма дозволяла також переглядати зображення в форматах: JPG, BMP, TIFF, GIF. Припустимо, що наступним ми хочемо додати підтримку форматі JPG. Ми розуміємо, що внесення змін до робочий продукт може привести до втрати функціоналу. Щоб, в разі виникнення проблем, не втрачати можливість перегляду зображень у форматі PNG, ми створюємо нову «гілку» dev, зміни якої не відіб'ються на «гілці» master. Після розробки і тестування нового функціоналу, в тому числі на сумісність з існуючим, ми проводимо «Злиття» «гілок» dev і master. Вся історія змін dev переноситься в master. Таким же чином ми поступово додаємо підтримку всіх потрібних нам форматів. Нижче пояснюються ключові терміни, які використовуються в статті [7].

- working copy - робоча (локальна) копія документів.
- repository, depot - сховище.
- revision - версія цього видання. Нові зміни (changeset) створюють нову ревізію сховища.
- check-in, commit, submit - фіксація змін.
- check-out, clone - витяг документа зі сховища і створення робочої копії.
- update, sync - синхронізація робочої копії до деякого заданого стану сховища (в тому числі і до більш старого стану, чемтекущее).
- merge, integration - злиття незалежних коммітов.
- conflict - ситуація, коли кілька користувачів зробили зміни одного і того ж ділянки документа.
- head - найсвіжіша версія (revision) в сховище.
- origin - ім'я головного сервера.
- branch (гілка) - напрям розробки проекту, незалежне відінших. Гілка є копією частини (як правило, одногокаталогу) сховища, в яку можна вносити свої зміни, які не впливають на інші гілки.

Документи в різних гілках мають однакову історію до точки розгалуження і різні після неї. Зміни з однієї гілки можна переносити в іншу.

Система одночасних версій (CVS):

CVS з'явилася в 1980-х і до сих пір популярна як у розробників комерційних продуктів, так і у open-source розробників. CVS поширюється на умовах Відкритого ліцензійної угоди GNU і дозволяє отримувати з сервера потрібну версію проекту - «check-out» (витяг), а потім пересилати назад на сервер, «check-in» (повернення), з внесеними змінами. Спочатку CVS була створена, щоб уникнути конфлікту версій. Всім учасникам для роботи надавалася тільки найостанніша версія коду. Це була перша система контролю версій. Користувачеві потрібно було швидко фіксувати зміни в репозиторії, поки інші не випередили його. Зараз CVS має підтримку роботи над проектами з гілками коду. Виходить кілька варіантів продукту з різними характеристиками, які можна буде об'єднати пізніше. Сервера CVS зазвичай працюють під управлінням Unix, але CVS-клієнти доступні і в інших популярних операційних системах. CVS - «зріла», перевірена часом система контролю версій. Це як і раніше опенсорсний система, але на сьогоднішній день нові функції додаються досить рідко. При цьому CVSNT, - виділилася в окремий проект версія CVS для серверів Windows, - зараз досить активно розширює функціонал.

<u>переваги:</u>

- Випробувана часом технологія, яка утримується на ринку десятки років.

<u>недоліки:</u>

- Перейменування або переміщення файлів не відбивається в історії;
- Ризики безпеки, пов'язані з символічними посиланнями на файли;
- Немає підтримки атомарних операцій, що може привести до пошкодження коду;

- Операції з гілками програмного коду дорогі, так як ця система контролю не призначена для довгострокових проектів з гілками коду;

Apache Subversion (SVN):

SVN створювалася як альтернатива CVS з метою виправити недоліки CVS і в той же час забезпечити високу сумісність з нею. Як і CVS, SVN це безкоштовна система контролю версій з відкритим вихідним кодом. З тією лише різницею, що поширюється під ліцензією Apache, а не під Відкритим ліцензійною угодою GNU. Для збереження цілісності бази даних SVN використовує так звані атомарні операції. З появою нової версії до фінального продукту застосовуються або все виправлення, або жодне з них. Таким чином, дані захищають від хаотичних часткових правок, які не узгоджуються між собою і викликають помилки. Багато розробники переключилися на SVN, так як нова технологія успадкувала кращі можливості CVS і в той же час розширила їх. У той час як в CVS операції з гілками метадати дорогі і не передбачені архітектурою системи, SVN створена якраз для цього. Тобто, для більших проектів з розгалуженням коду і багатьма напрямками розробки. Як недоліки SVN згадуються порівняно низька швидкість і брак розподіленого управління версіями. Розподілений контроль версій використовує пірингову модель, а не централізований сервер для зберігання оновлень програмного коду. І хоча пірінгова модель працює краще в open source проектах, вона не ідеальна в інших випадках. Недолік серверного підходу в тому, що коли сервер падає, то у клієнтів немає доступу до коду.

<u>переваги:</u>

- Система на основі CVS;
- Допускає атомарні операції;
- Операції з розгалуженням коду менш затратні;
- Широкий вибір плагінів IDE;
- Чи не використовує пірингову модель;

<u>недоліки:</u>

- Все ще зберігаються помилки, пов'язані з перейменуванням файлів і директорій;
- Незадовільний набір команд для роботи з репозиторієм;
- Порівняно невелика швидкість;

Git:

Ця система була створена для управління розробкою ядра Linux і використовує підхід, який в корені відрізняється від CVS і SVN. В основу Git закладалися концепції, покликані створити більш швидку розподілену систему контролю версій, на противагу регламентів та рішень, використаним в CVS. Так як Git розроблялася головним чином під Linux, то саме в цій ОС вона працює швидше за все. Git також працює на Unix-подібних системах (як MacOS), а для роботи на платформі Windows використовується пакет mSysGit.

Програмний код може бути недоступний, коли використовується комп'ютер без сховища. Для вирішення цієї проблеми є обхідні шляхи і деякі розробники вважають, що швидкодія Git є справедливою платою за незручності.

Крім того, в Git є безліч інструментів для навігації по історії змін. Кожна робоча копія вихідного коду містить всю історію розробки, що вкрай корисно, коли програмуєш без Інтернет-з'єднання.

<u>переваги:</u>

- Чудово підходить для тих, хто ненавидить CVS / SVN;
- Значне збільшення швидкодії;
- Дешеві операції з гілками коду;
- Повна історія розробки доступна оффлайн;
- Розподілена, пірінгова модель;

<u>недоліки:</u>

- Високий поріг входження (навчання) для тих, хто раніше використовував SVN;
- Обмежена підтримка Windows (в порівнянні з Linux);
- Mercurial;

- Mercurial була випущена одночасно з Git. Це також розподілена система контролю версій;

Mercurial :

Створювалася в якості альтернативи Git для розробки модулів ядра Linux. Але так як вибрали все-таки Git, то Mercurial використовується менше. Тим не менш, багато провідні розробники працюють саме з цією системою, наприклад OpenOffice.org. Система контролю версій Mercurial відрізняється від інших систем контролю версій тим, що головним чином вона написана на мові програмування Python. Однак, деякі частини виконані в якості модулів-розширень на C. Оскільки система децентралізована і написана на Python, багато Python-програмісти схиляються до переходу на Mercurial. Користувачі відзначають, що Mercurial зберегла деякі характеристики SVN, будучи при цьому розподіленою системою, і завдяки цій схожості, поріг входження в неї нижче для тих, хто вже знайомий з SVN. Також документація по Mercurial повніша, що допомагає швидше освоїтися з відмінностями. Один з істотних недоліків Mercurial полягає в тому, що на відміну від Git в ній не можна об'єднати дві батьківські гілки, так як Mercurial використовує систему плагінів, а не підтримку скриптів. Це відмінно підходить для деяких програмістів, але багато хто не хоче відмовлятися від можливостей Git.

<u>переваги:</u>

- У порівнянні з Git легше в освоєнні;
- Детальна документація;
- Розподілена модель системи контролю версій;

<u>недоліки:</u>

- Немає можливості злиття двох батьківських гілок;
- Використання плагінів, а не скриптів;
- Менше можливостей для нестандартних рішень;

Порівняння

Для порівняльного аналізу систем контролю версій скористаємося експертним методом оцінювання, результати якого зведені в таблицю 1.

Таблиця 1.1 - Порівняльний аналіз систем контролю версій.

Назва	**Швидкість**	**Тип истеми**	**функціональні можливості** Копіювання і перенесення файлів / папок, дослідження через підрядник)	**документація**	**Підтримуючі платформи** Unix, Windows, IOS)	**Ціна**
CVS	1 бал	0 балів	1 бал	1 бал	3 бала	1 бал
SVN	2 бала	0 балів	2 бала	2 бала	3 бала	1 бал
Git	5 балів	1 бал	3 бала	5 балів	3 бала	1 бал
Mercurial	3 бали	1 бал	3 бала	3 бала	3 бала	1 бал

Швидкість системи оцінюємо по 5-бальній шкалі. згідно з тестами продуктивності систем контролю версій, Git є найбільш швидкою з них. Потім йде Mercurial, а за ним SVN і CVS. На нашу думку, распредёленная система має ряд переваг перед централізованою. По-перше, не вимагає постійного підключення до мережі. По-друге, більш надійна з точки зору зберігання даних. При централізованому підході в разі порушення цілісності даних на сервері дані будуть втрачені, чого не трапиться при розподіленому підході, так як користувачі збережуть локальну копію сховища. В-третє, зручність роботи з декількома репозиторіями в проекті при міжнародний науковий журнал «Синергія наук» розподіленому підході. І тому ми оцінюємо системи з цим типом більш високим балом.У функціональні можливості ми оцінюємо підтримку можливостей [8]:

— копіювання файлів і папок;

— перенесення файлів і папок;

— відстеження змін в файлі через підрядник.

Максимальний бал — 3. CVS дозволяє тільки відслідковувати зміни в файлі і, відповідно, отримує 1 бал. SVN також через підрядник відстежує зміни,

а ще дозволяє переносити файли і папки. Git та Mercurial підтримують всі три можливості. Документацію ми оцінювали з точки зору новачка, тобто людини,

у якого ніколи не було досвіду роботи з даною системою. чим очевидніше спосіб отримання доступу до документації, чим більш зрозумілою мовою вона написана, тим більше балів отримує система. найбільш зручними виявилися Git і Mercurial. На їх офіційних сайтах прямо на головних сторінках є посилання на документації. Самі документації написані чітко і ясно. Що стосується CVS і SVN, то їх документації орієнтовані на фахівців. Людині без досвіду в них важко розібратися. Кросплатформеність є характерною рисою для всіх систем, тому все отримують максимальний бал. Ціна. Всі системи безкоштовні і отримують по одному балу. В результаті Git набирає найбільшу кількість балів. Ми рекомендуємо використовувати цю систему контролю версій. Але якщо у вас є вагома причина використовувати централізовану систему, то ми радимо вам

вибрати SVN.

1.4 Проблема які вирішить система контроля версій в хмарних системах до захисту персональних даних.

Персональні дані – це будь-яка інформація, що відноситься до ідентифікованої фізичної особи (суб'єкт даних), по якій прямо або опосередковано можна її визначити. До такої інформації можна віднести ім'я, дані про місцезнаходження, онлайн ідентифікатор, один або декілька факторів, характерних для фізичної, генетичної, розумової, економічної, культурної і соціальної ідентичності цієї фізичної особи. Визначення широке і достатньо чітко дає зрозуміти, що навіть IP-адреси користувачів також можуть бути персональними даними.Важливо відмітити, що існують деякі типи даних, що відносяться до категорії особливих або конфіденційних персональних даних. Це інформація, що містить: расове або етнічне походження, політичні погляди, релігійні або філософські переконання та членство в профспілках. Окрім того, до цієї групи відносяться генетичні та біометричні дані. Також корисним буде

заздалегідь продумати механізми реагування на запити користувачів і суб'єктів персональних даних, що можливі в рамках CRM систем, наприклад про уточнення даних, їх видалення, припинення обробки чи передачі іншій компанії в рамках права на переміщення даних. Загальний підхід до обробки персональних даних викладений у вигляді 6 основних принципів:

1. **Законність, справедливість та прозорість.** Персональні дані повинні оброблятися законно, справедливо та прозоро. Будь-яку інформацію про мету, методи та обсяги обробки персональних даних мається викладати максимально доступно та просто.
2. **Обмеження застосування.** Персональні дані повинні збиратися та використовуватись виключно з метою, що була заявлена компанією (або онлайн-сервісом).
3. **Мінімізація даних.** Забороняється збирати особисті дані в більшому обсязі, ніж той, що потрібен для досягнення мети обробки.
4. **Точність.** Особисті дані, які являються неточними, повинні бути видалені або виправлені (за вимогою користувача).
5. **Обмеження зберігання.** Особисті дані повинні зберігатися у формі, яка дозволяє ідентифікувати суб'єкти даних на строк не більше, ніж це необхідно для досягнення мети обробки
6. **Цілісність та конфіденційність.** При обробці даних користувачів компанія зобов'язана забезпечити захист персональних даних від несанкціонованої або незаконної обробки, знищення та пошкодження.

Основні вимоги до розробленої версії системи контролю версій

Вимоги SCV можна розсортувати за такими основними темами:

- **Контроль даних**

CRM очікує, що організації зможуть контролювати свої дані, щоб забезпечити доступ до них та обробку їх авторизованими користувачами лише

за необхідності. Вимоги до контролю наведені у статтях 5, 25 та 32 GDPR. Відповідно до вимог SCV організації повинні:

- передавати дані тільки авторизованим особам;
- забезпечити точність та цілісність даних;
- мінімізувати розкриття особистості суб'єкта;
- застосовувати заходи із убезпечення даних;

Шифрування – це саме той метод, при якому дані знаходяться в нечитабельному вигляді, якщо тільки користувач, або процес не надасть відповідний ключ. Відповідно до CRM, цей простий метод може забезпечити доступ до персональних даних тільки авторизованим користувачам, а також контролювати час, протягом якого цей доступ може бути здійснений.

- **Безпека даних**

SCV гарантує конфіденційність. Обов'язки з захисту інформації регулюються статтями 6, 25, 28 та 32. Для збереження приватності суб'єкта організаціям необхідно:

- застосувати захист даних за проектом та за замовчуванням;
- включити безпеку до зобов'язань за контрактом з партнерами та постачальниками послуг;
- використовувати шифрування або псевдонімізацію;
- застосувати заходи безпеки, що стосуються оцінки ризиків;
- вжити заходів, якщо вони зберігають дані для подальшої обробки;

CRM спеціально вказує на шифрування, як на основну вимогу безпеки даних. Крім того, організаціям потрібно провести оцінку ризиків, а потім прийняти заходи, що пом'якшують ризики, які вони виявлять. Оскільки жодна організація не може повністю визначити або передбачити всі ризики для своїх даних, і жоден підхід до периметра безпеки не є надійним, організації повинні шифрувати свої дані, щоб забезпечити відповідність до CRM. За допомогою шифрування неважливо, чи є порушення – дані будуть належно захищені.

- **Право видалення**

Навіть після того, як дані вже зібрані, суб'єкти даних все одно мають певний контроль над цими даними. "Право на видалення" розглядається у статтях 17 та 28. CRM вимагає від організацій повністю видалити дані з усіх сховищ в разі, якщо:

- користувач відкликає свої дані;

- партнерська організація вимагає видалення персональних даних своїх користувачів, що були наданні для обробки;

- термін дії угоди про використання персональних даних скінчився;

Згідно цієї норми, передбачаються випадки, коли організація повинна буде забезпечити повне видалення персональних даних своїх користувачів. Це дуже складна вимога тому, що навіть після видалення, дані все-одно зберігаються на магнітних носіях інформації. Але щоб повністю відповісти нормі, організації можуть шифрувати дані, а потім видалити тільки ключ шифрування. Цей метод перетворює дані в повністю і назавжди нечитабельний масив інформації.

- **Профілактика ризиків та комплексна перевірка**

Організації повинні самостійно оцінювати ризики, що пов’язані з конфіденційністю та безпекою даних, а також демонструвати, що вони вживають всіх відповідних заходів з урахуванням зроблених ними висновків. Ці обов'язки викладені у статтях 2, 24 та 28. З метою зменшення ризиків та виконання комплексної перевірки організація зобов’язана:

- виконати повну оцінку ризиків;
- застосувати заходи для забезпечення та демонстрації відповідності;
- активно сприяти партнерам та клієнтам у досягненні відповідності;
- продемонструвати повний контроль над даними;

Коли організація укладає контракти з партнером чи стороннім сервісом, вони не відмовляються від відповідальності за безпеку даних. Фактично, організаціям буде покладено договірне зобов'язання допомагати один одному стосовно безпеки даних та мінімізації ризиків. Оскільки шифрування забезпечує безпеку безпосередньо до даних, то організації-партнери залишаються відповідними даній вимозі, оскільки захист гарантується незалежно від того на чиїй стороні зараз знаходяться персональні дані.

- **Повідомлення про витік даних**

Коли порушення безпеки загрожує правам та конфіденційності суб'єкта даних, організаціям необхідно повідомити клієнтів та їх наглядовий орган. Обов'язки щодо повідомлення про порушення викладені у статтях 33 та 34. В рамках CRM організації зобов'язані:

- повідомити свій наглядовий орган протягом 72 годин;
- описати наслідки порушення безпеки даних;
- повідомити про порушення безпосередньо суб'єктів даних;

Висновки до розділу 1

Розглянуто принцип роботи сучасних хмарних систем, управління даними в хмарних системах. Огляд основних характеристик хмарних обчислень, види архітектурних рішень для хмарних систем, що є важливим питанням, якщо потрібно налаштовувати системи контролю версій. Наведено детальний огляд моделі організації мультітенантного сховища даних, так як запропоновані в роботі рішення базуються на системах даної моделі, де застосунок володіє вбудованими можливостями для обслуговування декількох груп користувачів, які можуть знищити, переписати дані системи, які вплинуть на подальшу роботу системи в цілому. Виконано огляд принципів захисту даних та ізоляції від конкретних користувачів системи. Було зроблено опис сучасних готових рішень та продуктів систем контролю версіх та їх порівняння. Порівнюючи переваги та недоліки для кожного з них, побудовано порівняльну таблицю. Зроблено висновки для подальшої розробки власної системи контролю версій, а саме: пониження порогу входження, щоб користувач не витрачав багато часу для розуміння як користуватися даною розробкою, рішення питання ризиків безпеки, пов'язані з символічними посиланнями на файли та інші, які будуть детально описані в даній роботі. Поставлено конкретні вимоги для розбробки системи контолю версій.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО МЕТОД РОЗВ'ЯЗАННЯ, ЙОГО ОБГРУНТУВАННЯ, ПОРІВНЯННЯ ГОТОВИХ РІШЕНЬ

2.1 Принципи роботи інструменту контролля версій

Інструмент система контролю версій який являє програмне забезпечення, що дозволяє створювати версії елементів і працювати з цими версіями, як з самостійними елементами. Робота з версіями передбачає як створення самих версій, так і структури для їх зберігання. Як правило, це або ланцюжка, або дерева.

Перш ніж працювати з елементами і їх версіями, треба ці елементи створити, тобто дати вказати системі контролю версій взяти наявні об'єкти реального світу і помістити їх під свій контроль. Разом з самим елементом завжди створюється і його перша версія.

Найчастіше в якості елементів для контролю версій виступають:

- файли;
- директорії;
- hard- і softlinks.

Усередині системи контролю самі елементи можуть розміщуватися по-різному - це залежить від архітектори VCS. Користувачеві важливо лише знати, що була розташована всередині сховища і робота з ним йде за допомогою команд обраного інструментарію.

1. Зберігання версії. етапи розробки. Зробивши зміни в один або кілька файлів проекту, програміст записує зміни в репозиторій - сховище всіх версій і змін проекту. Варто відзначити, що зберігається не весь проект цілком, з метою економії місця і часу збереження змін, в репозиторій додаються тільки файли, які зазнали змін (сервер з репозиторієм може бути віддаленим, і, якщо проект - дуже великий, то потрібно досить великий час для передачі всіх його файлів по мережі).

2. Оновлення до останньої версії розробленого програмного забезпечення. Так як, зазвичай, над розробкою проектів працює ціла команда фахівців, то вони постійно додають в репозиторій змінені файли. Тому одним з основних завдань системи контролю версій є можливість відслідковувати всі ці зміни і швидко оновлювати програмне забезпечення клієнтів до актуальної версії.

3. Об'єднання змін. Часто кілька програмістів одночасно змінюють одні і ті ж файли. Якщо зміни не перетинаються, то системи контролю версій дозволяють легко і просто об'єднати ці зміни.

4. Вирішення конфліктів. Якщо кілька людей змінили один і той же ділянку коду, то, автоматично, об'єднати такі зміни - неможливо. Зазвичай, системи контролю версій надають собою інструменти, що дозволяють вручну додати необхідні правки в тест програм, щоб об'єднати конфліктуючі частини коду.

5. Відкочування до попередніх версій. Якщо обраний напрям у розвитку проекту виявилося тупиковим або містить помилки, то системи контролю версій дозволяють повернути розроблене програмне забезпечення до однієї з останньої робочої версії, просто, скопіювавши з репозиторію потрібну версію програмного забезпечення, або окремі файли.

6. Супровід декількох напрямків розвитку програмного забезпечення. Не завжди можна відразу зберігати внесені зміни. Часто доводиться досить довго розробляти і налагоджувати окремі правки перш, ніж їх можна об'єднати з основним програмним забезпеченням. У цьому випадку багато систем контролю версій дозволяють організовувати паралельні гілки з контролю декількох напрямків розвитку програмного забезпечення, швидко перемикатися між ними, а потім об'єднувати їх в єдине ціле.

2.2 Види архітектури системи контролю версій

2.2.1 Локальні системи контролю версій

Багато людей в якості одного з методів контролю версій застосовують копіювання файлів в окрему директорію (можливо навіть директорію з відміткою за часом, якщо вони достатньо розумні). Даний підхід є дуже поширеним завдяки його простоті, проте він, неймовірним чином, схильний до появи помилок. Можна легко забути в якій директорії ви знаходитеся і випадково змінити не той файл або скопіювати не ті файли, які ви хотіли.

Щоб справитися з цією проблемою, програмісти давно розробили локальні СКВ, що мають просту базу даних, яка зберігає всі зміни в файлах під контролем версій.

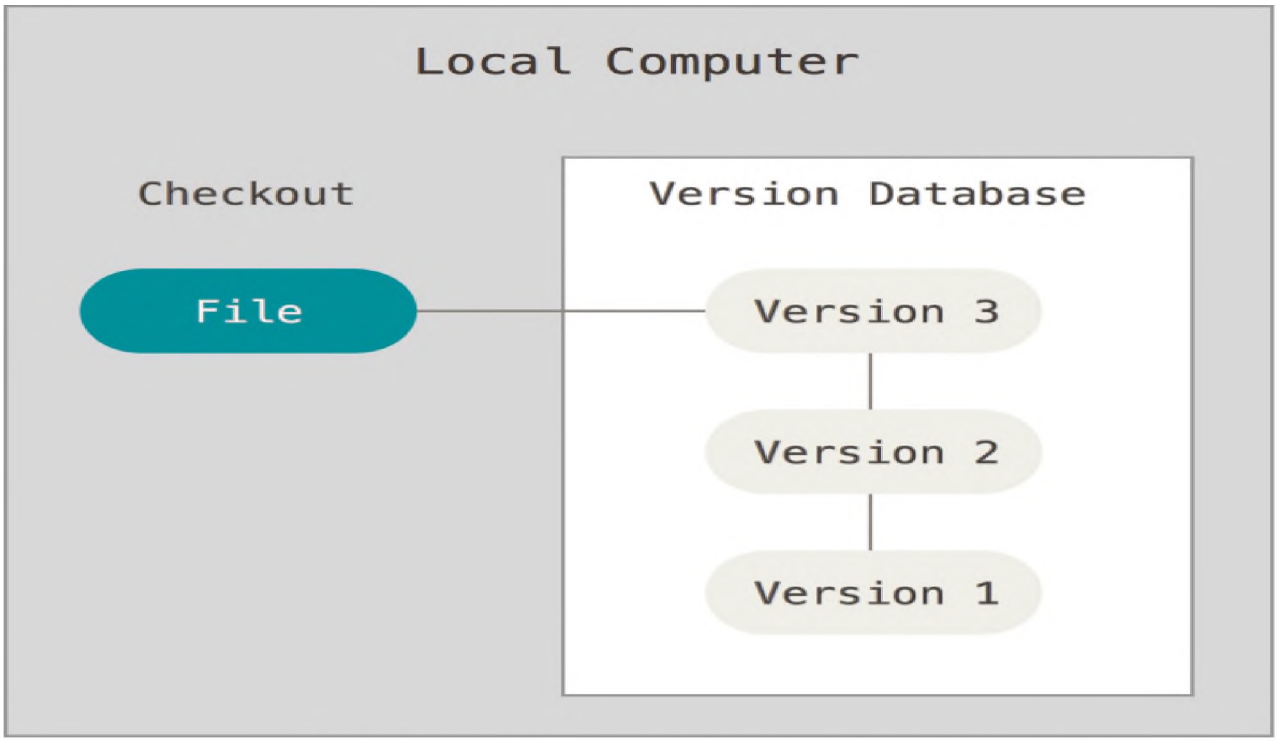

Рисунок — 3.1 Локальні системи контролю версій.

Одним з найбільш поширених інструментів СКВ була система під назвою RCS, яка досі поширюється з багатьма комп'ютерами сьогодні. RCS зберігає набори латок (тобто, відмінності між файлами) в спеціальному форматі на диску; він може заново відтворити будь-який файл, як він виглядав, в будь-який момент часу, шляхом додавання всіх латок.

2.2.3 Централізовані системи контролю версій

Наступним важливим питанням, з яким стикаються люди, є необхідність співпрацювати з іншими розробниками. Щоб справитися з цією проблемою, були розроблені централізовані системи контролю версій (ЦСКВ). Такі системи як CVS, Subversion і Perforce, мають єдиний сервер, який містить всі версії файлів, та деяке число клієнтів, які отримують файли з центрального місця. Протягом багатьох років, це було стандартом для систем контролю версій.

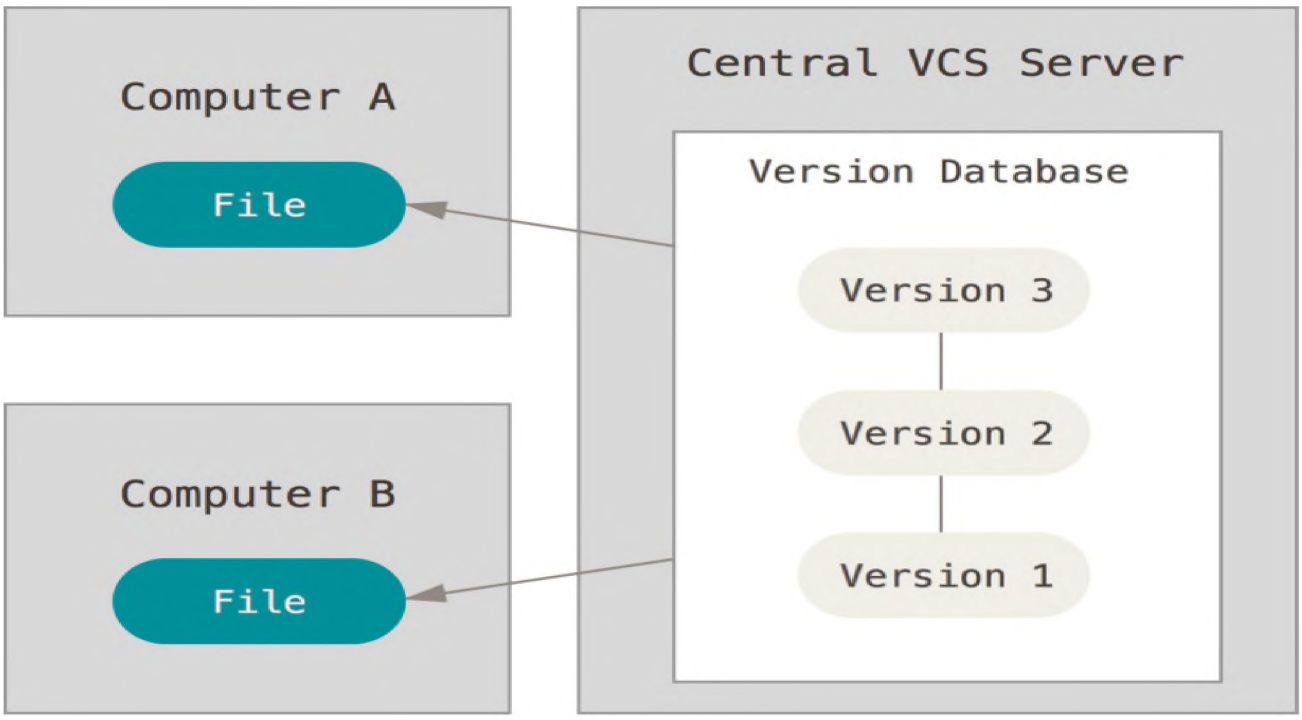

Рисунок — 3.2 Централізовані системи контролю версій.

Такий підхід має безліч переваг, особливо над локальними СКВ. Наприклад, кожному учаснику проекту відомо, певною мірою, чим займаються інші. Адміністратори мають повний контроль над тим, хто і що може робити. Набагато легше адмініструвати ЦСКВ, ніж мати справу з локальними базами даних для кожного клієнта.

Але цей підхід також має деякі серйозні недоліки. Найбільш очевидним є єдина точка відмови, яким є централізований сервер. Якщо сервер виходить з ладу протягом години, то протягом цієї години ніхто не може співпрацювати або зберігати зміни над якими вони працюють під версійним контролем. Якщо жорсткий диск центральної бази даних на сервері пошкоджено, і своєчасні

резервні копії не були зроблені, ви втрачаєте абсолютно все — всю історію проекту, крім одиночних знімків проекту, що збереглися на локальних машинах людей. Локальні СКВ страждають тією ж проблемою — щоразу, коли вся історія проекту зберігається в одному місці, ви ризикуєте втратити все.

2.2.3 Децентралізовані системи контролю версій

Долучаються до гри децентралізовані системи контролю версій (ДСКВ). В ДСКВ (таких як, Git, Mercurial, Bazaar або Darcs), клієнти не просто отримують останній знімок файлів репозиторія: натомість вони є повною копією сховища разом з усією його історією. Таким чином, якщо вмирає який-небудь сервер, через який співпрацюють розробники, будь-який з клієнтських репозиторіїв може бути скопійований назад до серверу, щоб відновити його. Кожна копія дійсно є повною резервною копією всіх даних.

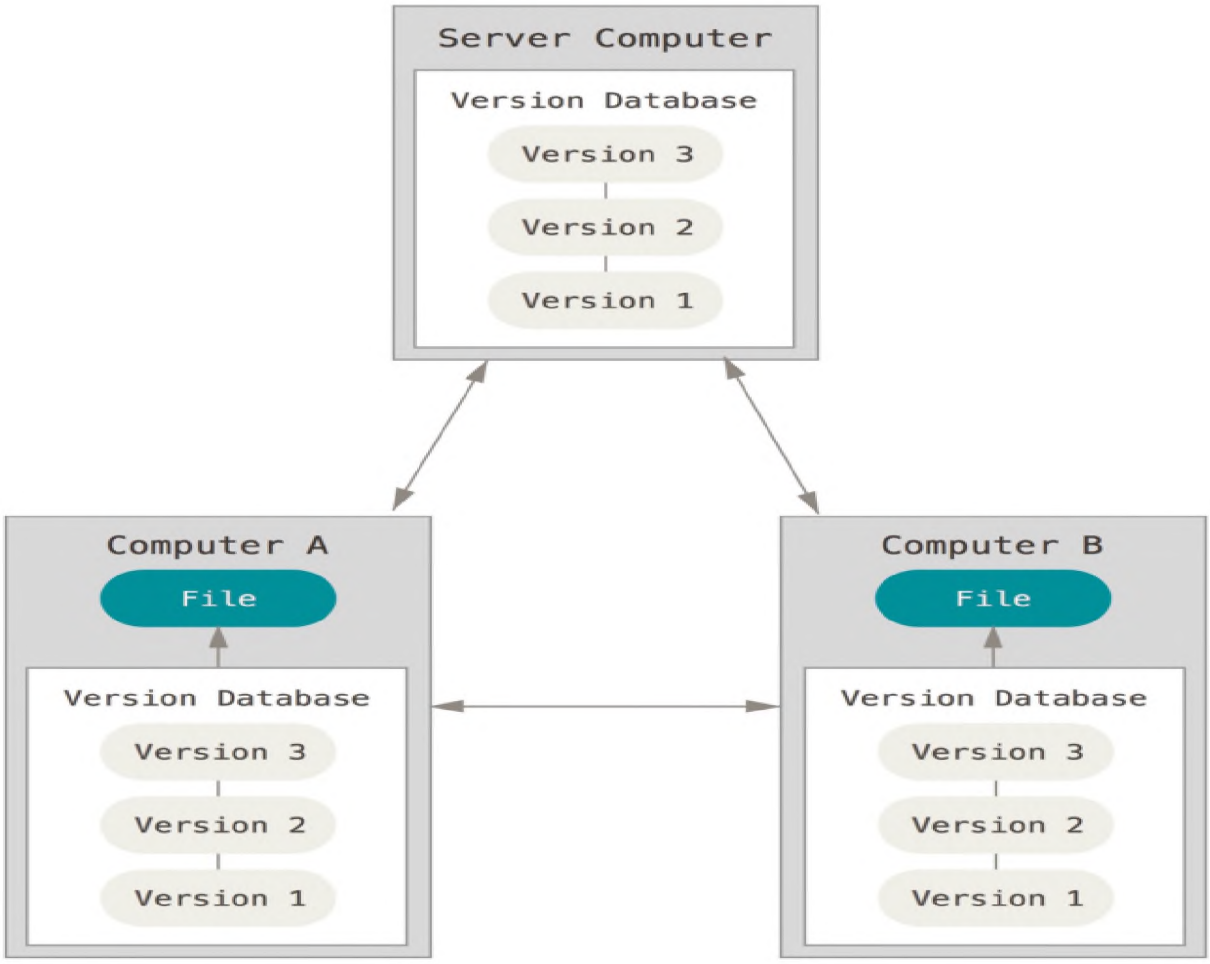

Рисунок — 3.3 Децентралізовані системи контролю версій.

Більш того, багато з цих систем дуже добре взаємодіють з декількома віддаленими репозиторіями, так що ви можете співпрацювати з різними групами людей, застосовуючи різні підходи в межах одного проекту одночасно. Це дозволяє налаштувати декілька типів робочих процесів, таких як ієрархічні моделі, які неможливі в централізованих системах.

2.3 Використання протоколу OAuth 2.0 в SalesForce.

OAuth - відкритий протокол, що забезпечує простий, стандартний спосіб для безпечної авторизації з додатків через інтерфейс API. OAuth дозволяє надати додаткам доступ до ресурсів, не розкриваючи додатків реєстраційні дані користувачів. Програми, що використовують OAuth, можуть пройти перевірку справжності і отримати доступ до ресурсів SalesForce навіть під час відсутності користувача. Платформа SalesForce реалізує інфраструктуру авторизації OAuth 2.0. API-інтерфейси, такі як API SalesForce REST і API веб-служби SOAP, а також Chatter REST API, можуть використовувати OAuth 2.0 для авторизації з метою представлення доступу до ресурсів SalesForce. Таким чином я отримую доступ до файлів які генерую при кожному деплої інформації в систему.

2.3.1 Метод застосування OAuth 2.0 в SalesForce

Протокол OAuth чимось нагадує ключ паркувальника, який надає обмежений доступ до автомобіля. Такий ключ дає можливість керувати автомобілем, але не дозволяє відкрити багажник або бардачок. Аналогічно OAuth надає клієнтського додатку обмежений доступ до призначених для користувача даних на сервері ресурсів. Коли клієнтську програму відправляє серверу авторизації запит на авторизацію, той видає додатком маркер.

Можна також вважати OAuth зв'язком між веб-сайтом з фотографіями і послугою друку, як описано в OAuth 2.0 RFC. Наприклад, Flickr, сайт з фотографіями - це сервер ресурсу, а Snapfish, служба друку - клієнт. Snapfish отримує на обмежений час доступ до певних фотографій на Flickr «тільки для читання», а потім авторизація Snapfish закінчується і стає недійсною. Можна

також вказати серверу авторизації відкликати маркер доступу Snapfish. Відгук усуває доступ, якщо ви не довіряєте сервера припинити його самостійно.

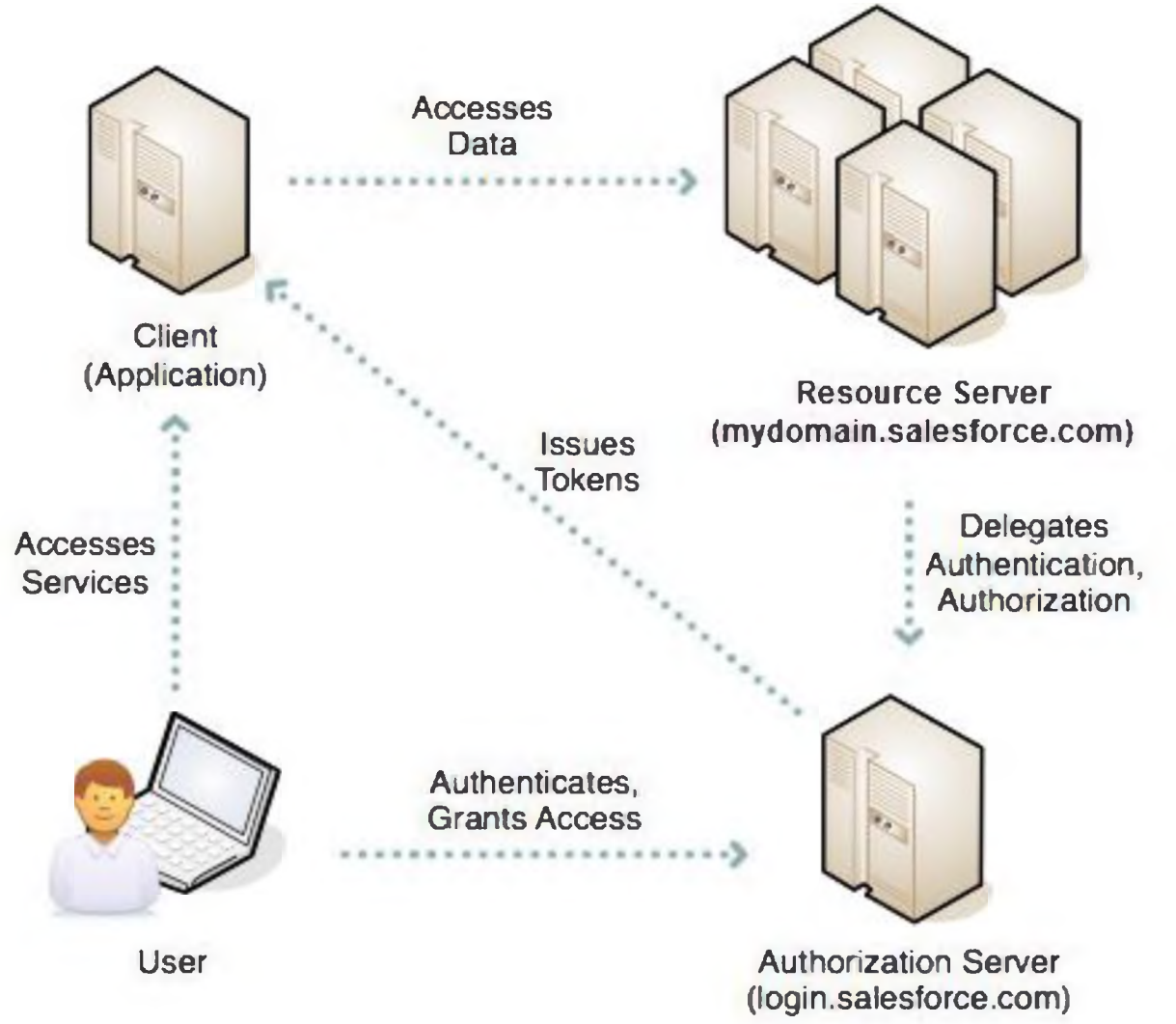

Рисунок — 3.4. Відображення роботи OAuth 2.0 в системі SalesForce

Така авторизація помітно відрізняється від випадку, коли клієнтського додатку повідомляється ім'я користувача і пароль користувача для доступу до сервера ресурсів. Додаток клієнта діє від вашого імені, маючи такий же доступ до даних. Якщо користувач перестає довіряти клієнтського додатку, необхідно змінити пароль на сервері ресурсів, а це створює незручність і загрозу безпеці. Тому використання маркерів OAuth - більш оптимальне рішення.

2.3.2 Процес перевірки автентичності маркера носія OAuth 2.0 JWT

Процес перевірки автентичності маркера носія OAuth 2.0 JWT визначає спосіб використання JSON Web Token (JWT) для відправки запиту на отримання маркера доступу OAuth від системи SalesForce в тому випадку, якщо клієнт вибирає попередню авторизацію. Перевірка справжності авторизованого додатки забезпечується цифровим підписом, застосованої до JWT. JWT дозволяє відкривати загальний доступ до особистих даних і даними безпеки на безпечних доменах.

Процес перевірки автентичності маркера носія OAuth 2.0 JWT аналогічний процесу перевірки справжності маркера поновлення всередині OAuth. JWT передається на кінцеву точку маркера OAuth за допомогою методу POST; кінцева точка обробляє JWT і формує access_token на основі попереднього затвердження програми. Може знадобитися попереднє затвердження при використанні профілів і наборів повноважень, якщо ваша політика пов'язаного додатка встановлена на «Користувачі, допущені адміністратором, попередньо авторизовані». Попередня авторизація також використовується з твердженням кінцевого користувача і випуском маркера поновлення, якщо політика пов'язаного додатка встановлена на «Всі користувачі можуть авторизуватися самостійно». Зверніть увагу, що параметр refresh_token може бути відсутнім в клієнті, а client_secret може не бути передано на кінцеву точку маркера. Процес носія JWT підтримує алгоритм RSA SHA256, який використовує завантажений сертифікат в якості секрету підписування.

Розмір сертифіката не повинен перевищувати 4 КБ. В іншому випадку використовуйте кодування DER для зменшення розміру файлу.

2.3.3 Етапи процесу перевірки справжності маркера носія OAuth 2.0 JWT:

1. Розробник створює або використовує поточний пов'язану програму і реєструє сертифікат X509. Даний сертифікат відповідає секретному

ключу додатки. Збереження пов'язаного додатка ініціює створення і призначення параметрів «Ключ користувача» (значення OAuth client_id) і «Секрет користувача» з додатком.

2. Розробник створює додаток, що створює JWT. JWT підписано секретним ключем, пов'язаним з сертифікатом, а пов'язане додатки використовує сертифікат для перевірки підпису.
3. JWT відправляється на кінцеву точку маркера https://login.salesforce.com/services/oauth2/token або https://acme.force.com/customers/services/oauth2/token (при реалізації спільноти), де acme.force. com / customers є URL-адресою спільноти.
4. Кінцева точка маркера перевіряє підпис за допомогою сертифіката, зареєстрованого розробником.
5. Кінцева точка маркера перевіряє адресата (aud), відправника (iss), дійсність (exp) і суб'єкт (sub) JWT.
6. Якщо JWT дійсний і користувач або адміністратор раніше авторизував додаток, SalesForce видає access_token.

2.3.4 Формування маркера носія JWT:

- Сформуйте заголовок JWT з наступним форматом: `{"alg": "RS256"}`.
- `Base64url` кодує заголовок JWT згідно із зазначеним за адресою `http://tools.ietf.org/html/rfc4648#page-7`. Результат буде схожий на `eyJhbGciOiJSUzI1NiJ9`.
- Створюю набір тверджень JSON для JWT з використанням параметрів iss, sub, aud і exp.

```
{
"iss":"3MVG99OxTyEMCQ3gNp2PjkqeZKxnmAiG1xV4oHh9A
KL_rSK.BoSVPGZHQukXnVjzRgSuQqGn75NL7yfkQcyy7",
"sub": "my@email.com",
"aud": "https://login.salesforce.com",
"exp": "1333685628"
}
```

- Кодування `Base64url` шифрує набір тверджень JSON без розривів рядків. наприклад:

```
eyJpc3MiOiAiM01WRzk5T3hUeUVNQ1EzZ05wMlBqa3FlWkt
4bm1BaUcxeFY0b0hoOUFLTF9yU0suQm9TVlBHWkhRdWtYblZqelJn
U3VRcUduNzVOTDd5ZmtRY3l5NyIsICJwcm4iOiAibXlAZW1haWwu
Y29tIiwgImF1ZCI6ICJodHRwczovL2xvZ2luLnNhbGVzZm9
yY2UuY29tIiwgImV4cCI6ICIxMzMzNjg1NjI4In0=
```

- Створіть рядок для закодованого заголовка JWT і закодованого набору тверджень JWT в наступному форматі.

- `encoded_JWT_Header + "." + encoded_JWT_Claims_Set.`

- Нижче виділений закодований заголовок JWT:

```
eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiAiM01WRzk5T3hUeUV
NQ1EzZ05wMlBqa3FlWkt4bm1BaUcxeFY0b0hoOUFLTF9yU0su
Qm9TVlBHWkhRdWtYblZqelJnU3VRcUduNzVOTDd5ZmtRY3l5
NyIsICJwcm4iOiAibXlAZW1haWwu
Y29tIiwgImF1ZCI6ICJodHRwczovL2xvZ2luLnNhbGVzZm9y
Y2UuY29tIiwgImV4cCI6ICIxMzMz
Njg1NjI4In0=
```

- Підписую підсумковий рядок за допомогою SHA256 і RSA.

- Створюю рядок на основі рядка з даного етапу в наступному форматі:

`existing_string + "." + base64_encoded_signature`

- Маркери носія JWT відправляються на кінцеву точку маркера за адресою https://login.salesforce.com/services/oauth2/token, https://test.salesforce.com/services/oauth2/token або https: //acme.force. com / customers / services / oauth2 / token (при використанні спільноти), де acme.force.com/customers є URL-адресою спільноти.

- Параметри, які є обов'язковими для передачі за допомогою методу POST:

`grant_type-urn: ietf: params: oauth: grant-type: jwt-bearer.`

`assertion` - маркер носія JWT.

- Додаткові стандартні параметри:

`format` - формат відповіді може бути заданий подібно до процесу OAuth за допомогою параметра token або HTTP-заголовка Accept.

`scope` - даний процес не підтримує параметр scope. Значення даного параметра формується параметрами scope з попередніх тверджень.

```
POST /services/oauth2/token HTTP/1.1
Host: login.example.com
Content-Type: application/x-www-form-urlencoded

grant_type= urn%3Aietf%3Aparams%3Aoauth%3Agrant-
type%3Ajwt-
bearer&assertion=eyJpc3MiOiAiM01WRz...[omitted for
brevity]...ZT
```

Висновки до розділу 2

В даному розділі було розглянуто основні принципи роботи ситеми контролю версій, виконано детальне обгрунтування аспектів роботи сучасних систем контролю версый. Роглянуто види архітектури системи контролю версій та їх порівняння, а саме для локальних, централізованих та децентралізованих систем. Розлянуто основний принцип збереження даних с таких системах, маніпуляція над ними, забезпечення цілісності та конфіденційності даних. Розлянуто використання протоколу OAuth 2.0 в SalesForce для розуміння, як зберігати дані, коли користувач буде робити якісь маніпуляції в системі, проведено дослідження всіх підводних каменів API системи SalesForce для подальшої реалізації та уникнення конфліктів. Розглянуті етапи процесу перевірки справжності маркера носія OAuth 2.0 JWT та формування маркера носія JWT. Для прозорості була зроблена покрокова інструкція, як користуватися системою за допомогою консольних команд, які використовуються при розробці системи.

3 ПРОЕКТУВАННЯ АРХІТЕКТУРНОГО РІШЕННЯ ІМПЛЕМНТАЦІЇ

3.1 Схема взаємодії модулів в системі SalesForce

В ході на роботи над магістерською дисертацією біло ознайомлено з патентною базою та з нацковими статтями, та доступними документаціями котрі описують системи контролю версій тобто існуючі систем, рішення. На основі запропонованих рішень було виделено функціональну архітектуру для впровадження систему контроля версій в CRM систему на прикладі SaleForce. Розлянемо запропоноване архітектурне рішення яке созражено на рисунку 3.1.

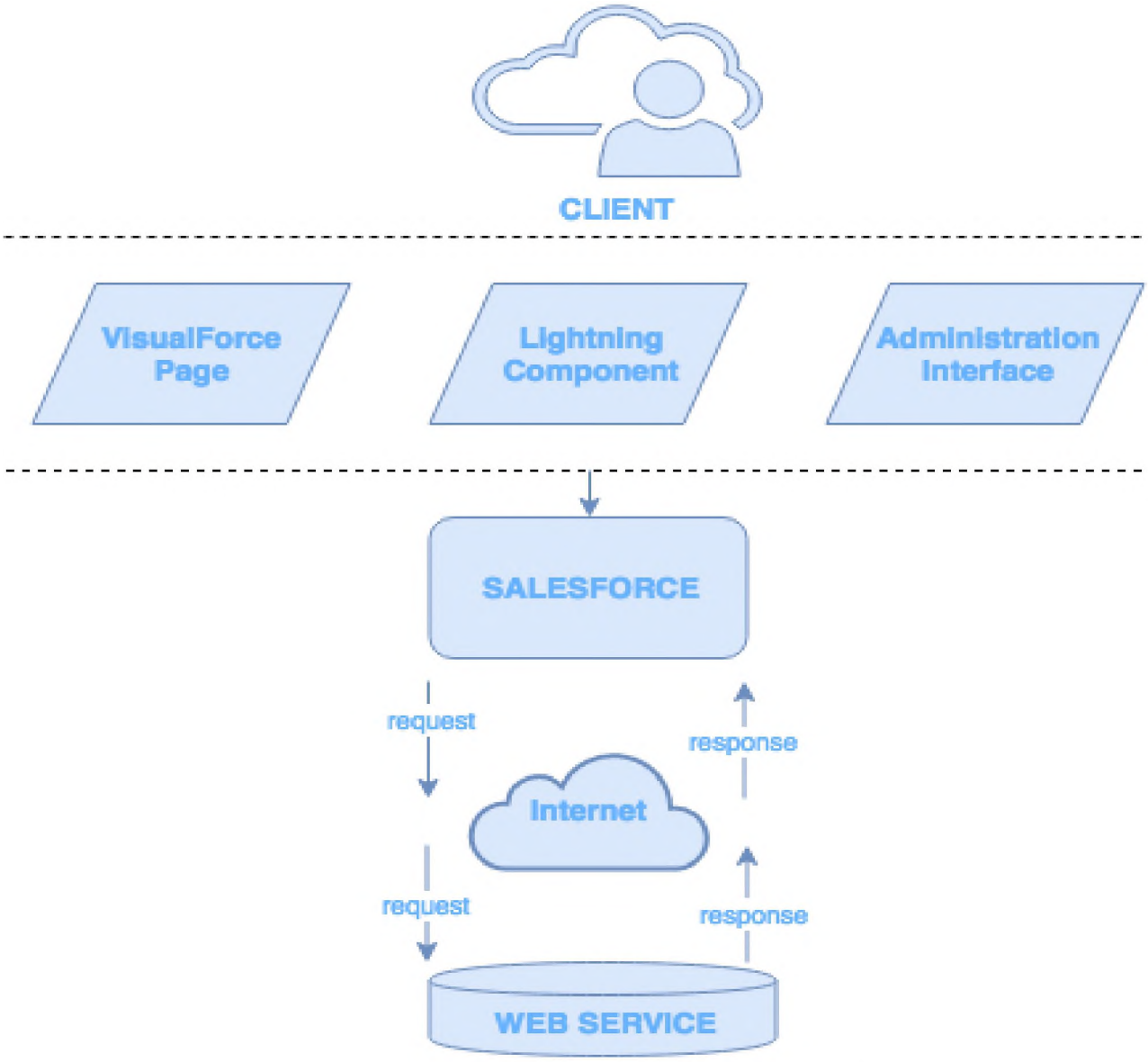

Рисунок — 3.1 Запропоноване архітектурне рішення

На рисунку 3.1 зображено архітектура яка відображає взаємодію клієнта з вебсервісом, клієнт робить маніпуляції використувуючи бруазер, консоль, публічний сайт написаний на сервісі SalesForce або власні компоненти через які можна зробити зміну файла в CRM системі. Клієн аутинтифікується в системі, після чого дає системі знати про зміни які він вніс, після чого система перехвачує файл зміни, оброляючи його в основній лозіці, архівує дані та зберігає в базу даних. Для чистоти розуміння розглянумо схематичне представлення коженого кроку обробки даних в запропонованій системі контролю версій.

Этап збереження даних

Під час змінину файлу який потрапляє під налаштування системи контроля версій, тобто є доступним в системі для перехоплення для подяльшого збереження а саме такі види файлів які було дослідженно системою. Такі файли як класи, структура обьетів яка має формат ".xml" файлу, Апекс класи, Тригери що мають власний формат, та інші типи які перечислено на рисунку 3.2.

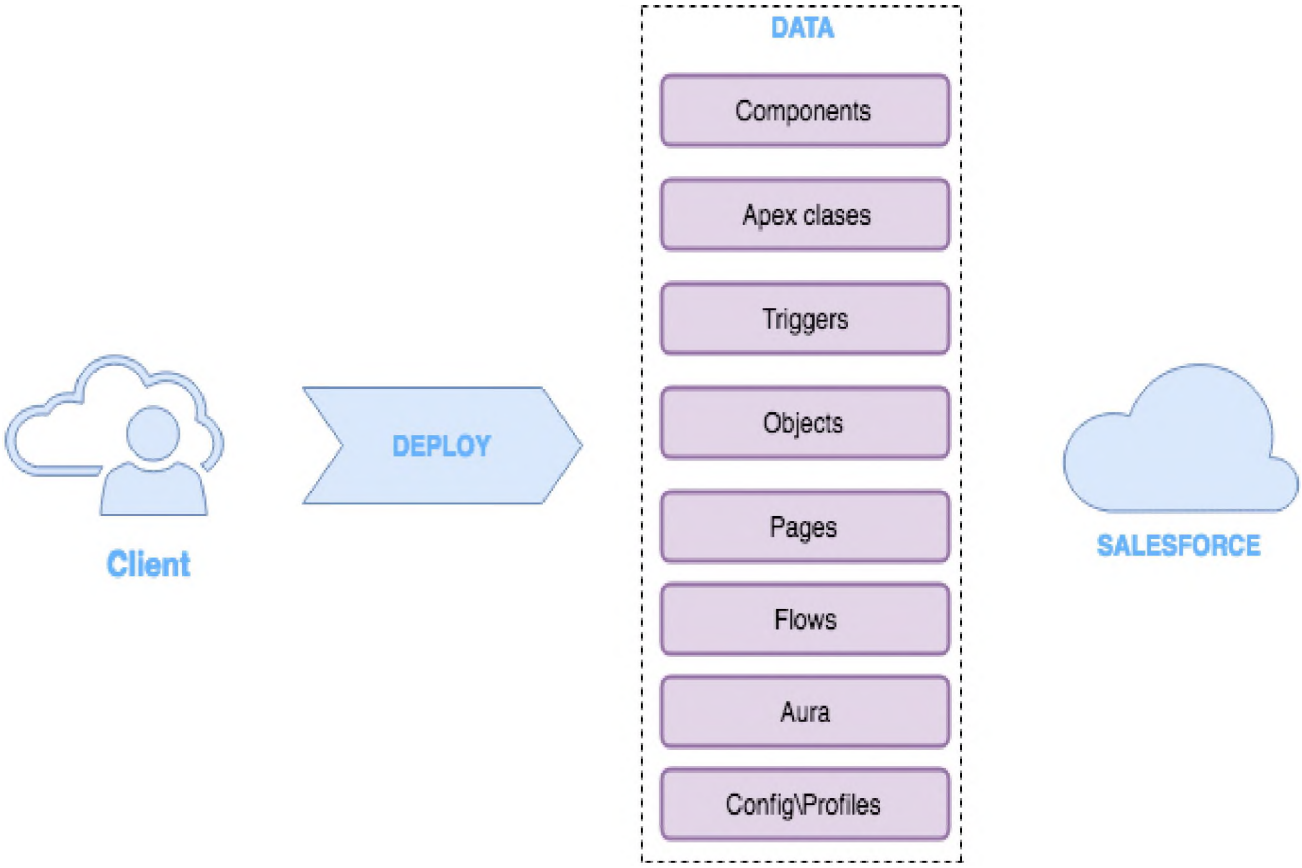

Рисунок — 3.2 Етап збереження даних в SalesForce систему

Користувач системи який зробив зміну фалу, має певну інформацію про себе, особистий профайл з налаштування доступу до конкретних даних системи, або це може бути груповий профайл який несе інформацію про групу людей в яку належить поточний користувач. Подалі змінені дані потрапляють в основну логіку обробки даних, але ще перед цим етапом може відбуватися етап авторизації та аутинтифікації клієта, дивлячись звідки відбуватиметься зміна даних, для такого кроку можуть потрапити такі шляхи:

- Публічний сайт (Community Portal, Napili, Public Site) , система сама опрацьовує етап авторизації та аутинтефікації, зачасту це користувач за замовчуванням так як система може навіть і не знати хто змінює дані, такий випадок зустрічається доволі рідко для вирішиння особливих випадків;
- Використання середовища роботи з системо, такі як Webstorm, Subline, intellij idea та інші в залежності від потреб користувача;
- Доступ до системи через зовнішню систему;

Етап обробки даних

Після успішної аутинтифіціх та авторизаціх користувача, його змінені дані перехоплюються запропонованою системою контролю версій. Основна логіка описана на Apex clases, за вимогами SalesForce, яка моніторить будь які зміни файлів які відслідковуються системою. Схематичне представлення бачимо на рисунку 3.3

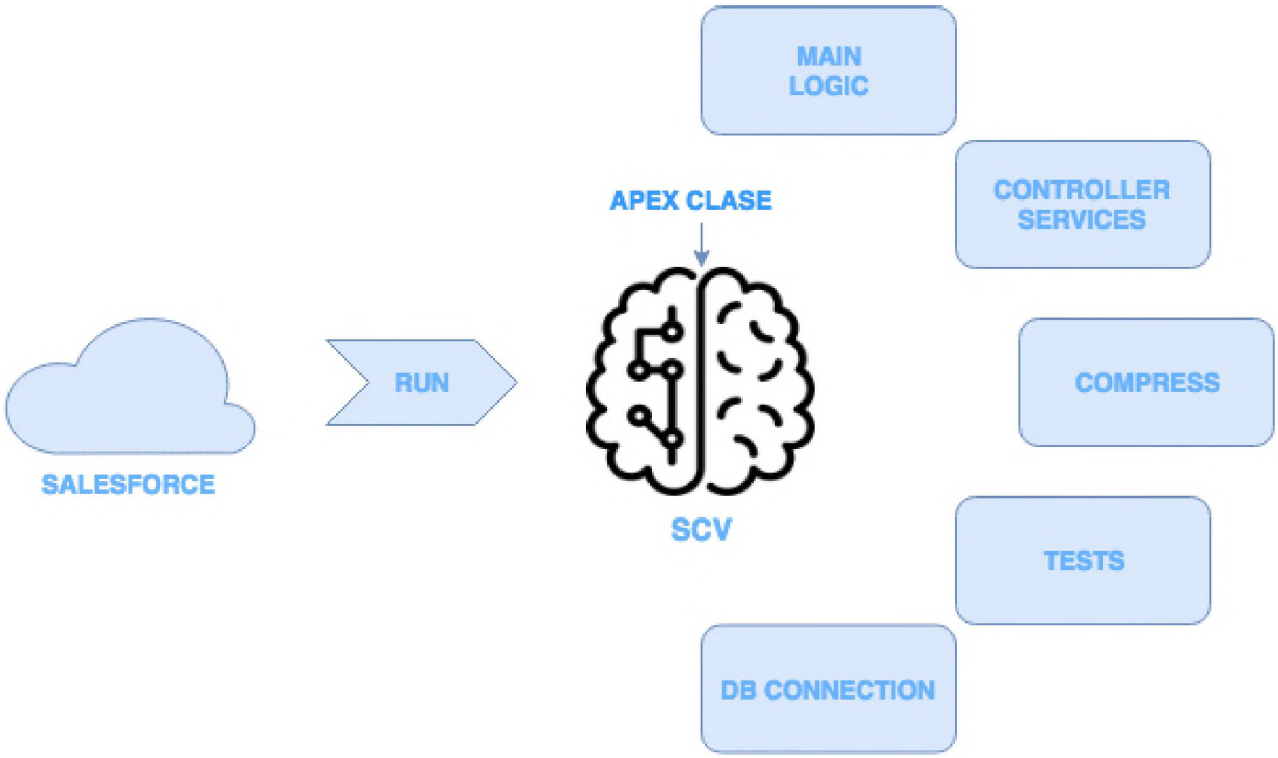

Рисунок — 3.3 Етап обробки даних перед збереженням

Можна помітити що система має основні блоки які виконують індивідуальну функцію. Кілька слів значимість блоку в розробці:

- Основна логіка влючає в себе збудження системи конntроля версій при будь-яких змінах файлів які спостерігаються, а саме внесення нового файлу в систему який до цього не існував, відновлення вже існуючого файлу та видалення існуючого файлу з системи;
- Блок стиснення даних охоплює в собі стиснення та зворотній процес - відновлення даних. Це потрібно для уникнення занадто великих файлів що не так сильно будуть заповнювати систему новими версіями в питані обьему. Для збереженя тіла файлу використано тип даних String Area що має сталий розмір 2кб, що є достатнім для збереження файлів та аудиту з урахуванням алгоритму стиснення;
- Доступ до бази даних налаштований до системи автоматично, тому щоб отримати доступ до бази даних мені потрібно користуватися правиласи системи та слідкувати за всіма лімітами та обмеженнями системи для уникнення помилок в процессі проботи;

- Блок тестування є доволі важливою частиною системи, існування даної реалізації дозволяє нам зібрати керуючий пакет який міститеме нашу систему контроля версій, що дає змогу іншим користувачам стороніх систем використовувати дану розробку для зручного користування системою чи в ласних потребах;
- Контролери та сервіси я будую за потребами “best practice” які описані в документації SalesForce. Використовую вже готові рішення з розробкою власнох логіки яка необхідна для реалізації процесі та основної ідеї даної розробки;

Даний етап є основним, мозком системи так як опрацьовую найважливіші процеси які виникають впроцесі роботи. Він є маштабованим та гнучким для подяльшої розробки.

Етап збереження даних.

Після успішної обробки даних які треба зберегти, використовуємо блок підключення до бази даних якив був описаний етапом вище. Маючи інформацію про користувача, тіло зміненого файлу, сворюю запис обьекту який був створений в системі з потреб розробки (буде описаний в подальшій роботі) в якому зберігаю посилання на попередню версію, унікальни ідетифікатор, ім’я файлу, власник файлу, посилання на користувача кто хробив зміни над файлом, дата коли було створено файл, та тіло файлу яку ми порівнюємо з попередніми та новими версіями файлу в процесі роботи. На рисунку 3.4 можно побачити схематичне представлення даного кроку обробки даних.

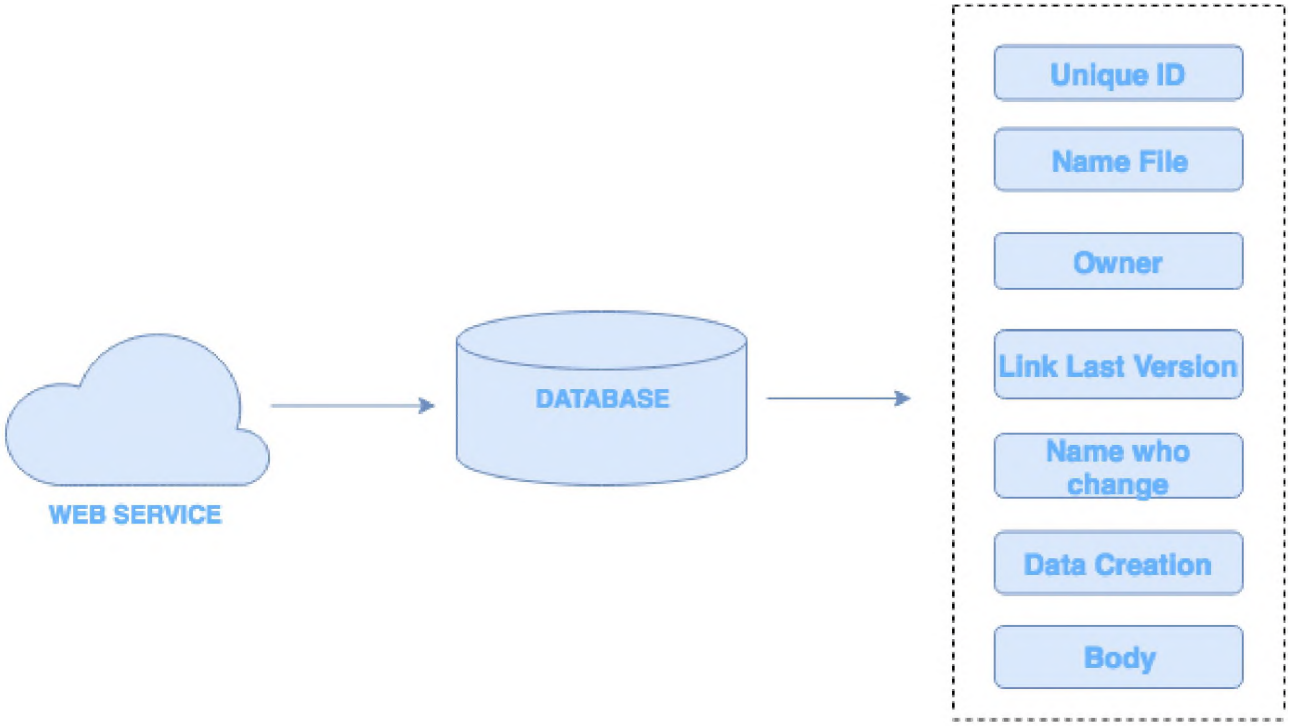

Рисунок — 3.4 Етап збереження даних в базу даних системи

Всі блоки обробки даних є тісно пов’язаними і не мають сенсу один без одного. В подальші розробці я буду описувати практичну частину даної реалізації і як її застосовувати в реальній системі.

Висновки до розділу 3

В даному розділі було розглянуто апробацію архітектурного рішення запропонованої розробки системи контролю версій. Запропонована система взаємодії розроблених модулів в системі SalesForce в цілому. Було зроблено схематичне представлення для кожного модюля, охарактеризовано основні можливості кожного з модулів та описаний поступой етап існування зміненого файлу в системі SaleForce. Описно головні можливості модулів та ключеві складові для кожного з них. Даний роздій е ключевим в питання реазізації програмного рішення та є опорою для подальшої розробки практичної частини та її застосування.

4 АПРОБАЦІЯ ЗАПРОПОНОВАНОГО АРХІТЕКТУРНОГО РІШЕННЯ КОНТРОЛЮ ВЕРСІЙ В СКЛАДІ SALESFORCE CRM СИСТЕМИ

4.1 Алгоритм стиснення даних

Дял реалізації систему контролю версій я вирішив скористатися алгоритмом стиснення даних що є процедурою перекодування даних, яку я проводитиму з метою зменшення обсягу, розміру, та об'єму файлів яки буду зберігати в системі. Стиснення базується на усуненні надлишку інформації, яка міститься у вихідних даних. Наприклад, повторення в тексті фрагментів (наприклад, слів природної або машинної мови). Подібний надлишок зазвичай усувається заміною повторюваних послідовностей коротшим значенням (кодом). Інший вид надлишковості пов'язаний з тим, що деякі значення в даних, що стискаються, трапляються частіше інших, при цьому можна замінювати дані, що часто трапляються, коротшими кодами, а ті, що рідко, довшими Стиснення даних, які не мають властивості надлишку неможливе. Також, зазвичай, неможливо стиснути зашифровану інформацію. Види стиснення даних існує два види:

- Стиснення без втрат - метод стиснення даних, при використанні якого закодована інформація може бути повністю відновлена зі стиснутих даних.
- Стиснення зі втратами - метод стиснення даних, при якому запакований файл відрізняється від оригіналу, проте може бути шкідливим для використання.

Так, як система контроля працює з файлами які потрібно вертати до оригінального стану від моменту його збереження, однозначний вибір використовувати стиснення без втрати даних.

З точки зору безпеки інформації – вдалий вибір стиснення даних впливає на цілісність інформації; а також опосередковано впливає на доступність (оскільки економить ресурси системи).

4.1.1 Використання LZ4 алгоритм стиснення та імплементація його в системі SalesForce.

LZ4 це універсальний алгоритм стиснення даних без втрат, пристосований для великої швидкості пакування та розпакування. Стиснення даних в методі LZ4 представляються у вигляді послідовності записів. Кожен запис починається з токена - одного байта, розбитого на два 4-х бітних поля. Перше поле визначає кількість байтів літеральної послідовності - тобто рядки, яка при розпакуванні буде скопійована в вихідний потік. Друге поле визначає довжину рядка, копируемой з уже розпакованого буфера (зі словника). Значення 0 в поле відповідає мінімальній довжині збіги в 4 байта. Значення 15 в поле є ознакою використання додаткового байта, значення якого буде додано до довжини. Якщо додатковий байт довжини дорівнює 255, то до поля довжини додається значення ще одного байта, що дозволяє вказувати довільні довжини через серію байтів із значенням 255 (0xff). Рядок литерала в стислій послідовності слід за токеном і додатковими байтами довжин литерала. Потім записується зміщення збіги у вихідному буфері та додаткові байти довжини збігу. Додатково можуть використовуватися фрейми, що вказують на розмір даних і містять контрольні суми. LZ4 являється доволі таки новим алгоритмом тому, тому я вирішив обрати його але при цьому зробив порівняння вже існуючих алгоритмів щоб остаточно бути впевненим у своєму виборі. Пердставляю таблицю порівннян існуючих алгоритмів:

Таблиця — 4.1 Порівняння існуючих рішень

Назва	Ступінь стиснення	Швидкість кодування	Швидкість декодування
zlib 1.2.8 -6	3.099	18	275
ZSTD	2.872	201	498

zlib 1.2.8 -1	2.730	58	250
LZ4 HC r127	2.720	26	1720
QuickLZ 1.5.1b6	2.237	323	373
LZO 2.06 2.106	2.106	351	510
Snappy 1.1.0	2.091	238	964
LZ4 r127	2.084	370	1590
LZF 3.6	2.077	220	502

У порівнянні з забезпечують рекордні показники системами, запропонований алгоритм не є однобоким (швидкість за рахунок ступеня стиснення або ступінь стиснення за рахунок швидкості) і забезпечує відмінне співвідношення швидкості та ефективності стиснення. Бібліотека з еталонною реалізацією алгоритму поширюється під ліцензією BSD. Примітною особливістю LZ4 також є можливість налаштування споживання пам'яті, що дозволяє використовувати LZ4 на вбудованих системах з невеликим розміром ОЗУ або на серверах, одночасно обробляють велику кількість стислих потоків. Для декодування необхідно заповнення таблиць трансформації, розмір яких може бути налаштований від 2.5 до 20 Кб, а також потрібно виділення пам'яті під буфер з вікном стиснення, який за замовчуванням становить 512 Кб, але може бути за бажанням зменшений до декількох кілобайт або збільшений до гігабайт (чим більше розмір вікна - тим вище рівень стиснення). У процесі стиснення даних додатково потрібно виділення пам'яті під буфер сортування, який за замовчуванням становить 128 Кб, але може бути довільно зменшений або збільшений. На наведеному нижче графіку відображені параметри експерименту зі стиснення файлу, передачі потоку по мережі і його розпакування яким ведеться розмова.

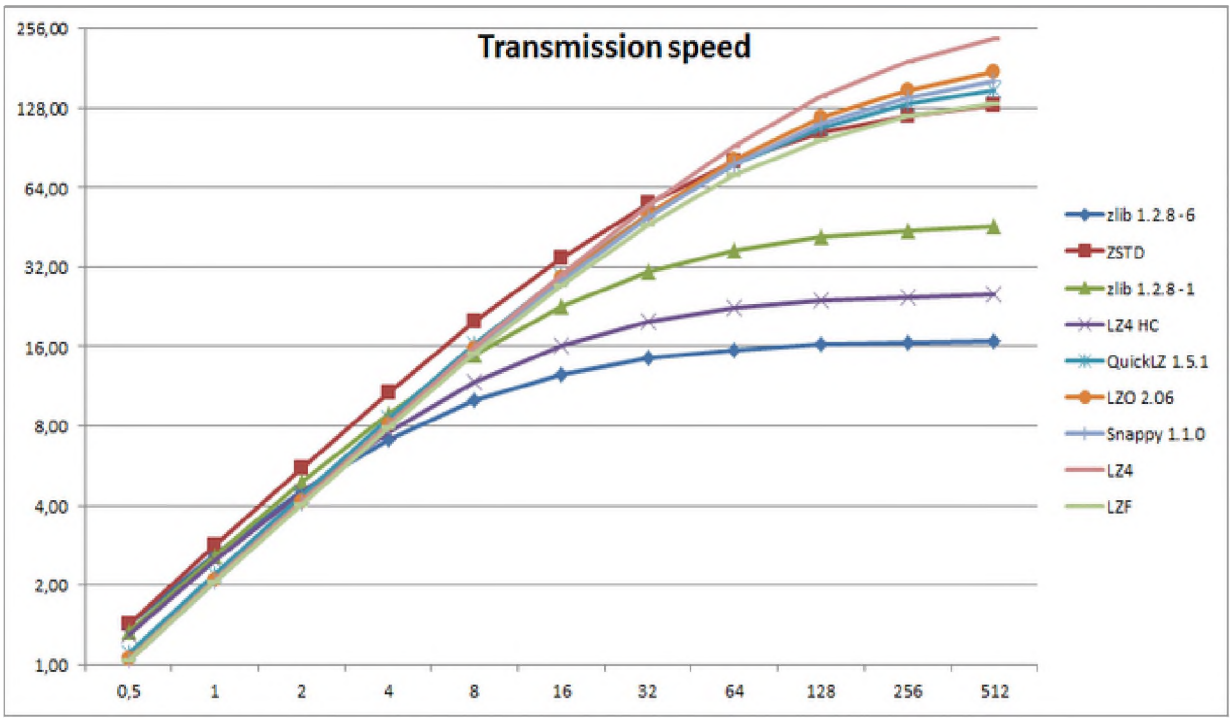

Рисунок — 4.1 Пропускна спроможність стиснення даних.

Перший графік показує співвідношення часу виконання операції (вісь Y) до пропускної спроможності каналу зв'язку в Мб / сек (вісь X).

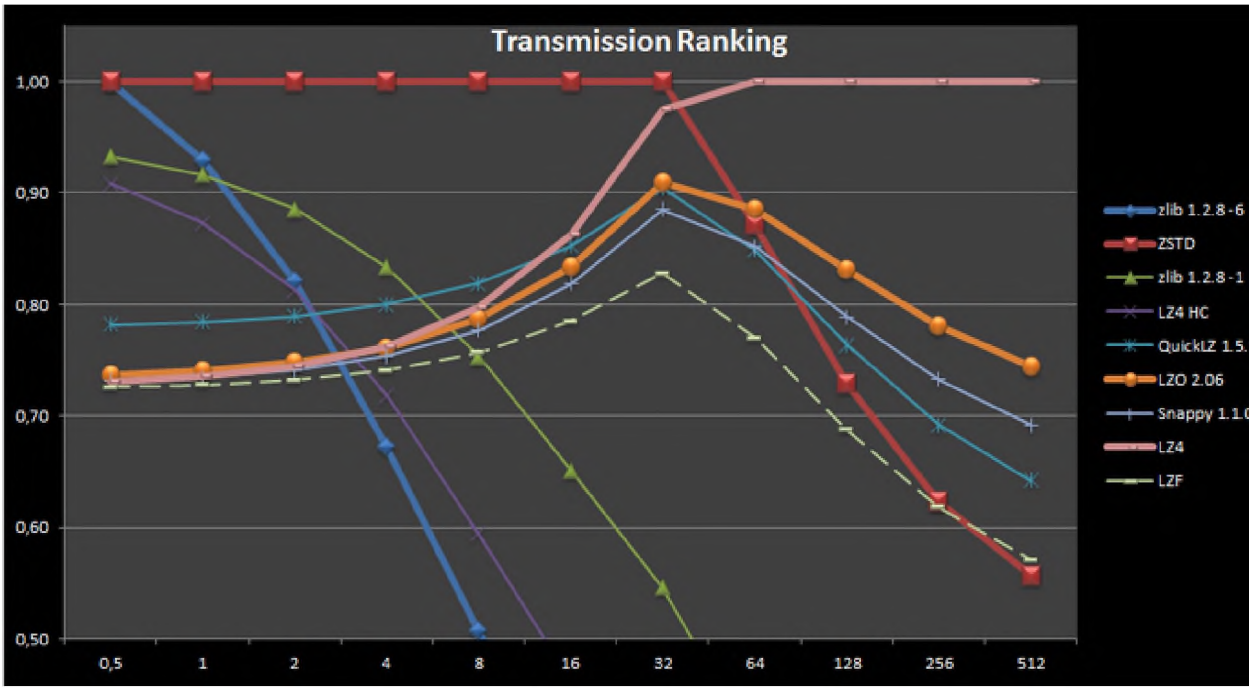

Рисунок — 3.2. Відношення часу до використаної пам'яті.

Другий графік відрізняється тим, що замість часу використовується відносне ранжирування алгоритмів, при якому за 1 прийнятий кращий

результат, а решта показників показані в процентному співвідношенні до нього. Таким чином за графіком видно, що LZ4 залишається лідером на швидкостях вище 50 Мб / сек, а ZSTD демонструє кращі результати на швидкостях від 0.5 Мб / сек до 50 Мб / сек.

4.2 Концепція системи збереження персональних даних на основі інструменту системи контроля версій

Можливості систем керування конфігурацією які пропонує моя розробка в CRM системі SalesForce:

1. Ідентифікація конфігурації – полягає в налаштуванні та підтримці базового стану системи, яка визначає архітектуру системи або підсистеми та її компонентів;
2. Керування конфігурацією – одна з основних функцій, яка включає в себе оцінку та контроль усіх запитів на змінення компонентів або налаштувань системи з ціллю подальшого утвердження або заборони цих змін;
3. Облік стану файлу конфігурації – функція яка записує стан елементів конфігурації і в випадку коли в системі з'явилася проблема, дає можливість повернути її до базової конфігурації і перевірених модифікацій;
4. Перевірка і аудит конфігурації – перегляд апаратного або програмного забезпечення для оцінки відповідності встановленим вимогам до продуктивності. Аудит перевіряє чи документація по конфігурації системи і підсистеми відповідає функціональним та фізичним характеристикам до прийняття в базову архітектуру.
5. Система керування конфігурацією повинна бути орієнтована на вирішення чотирьох основних завдань: визначення конфігурації, регулювання конфігурації, облік стану і перевірка якості конфігурації.
6. Основою функцією системи керування конфігурацією є контроль за внесенням змін до системи. Кожні змінення в системі повинні бути визначені та задокументовані. Також треба приймати заходи по підвищенню рівня безпеки систем керування конфігурацією з ціллю обмеження доступу зловмисників до

налаштувань системи або її відключення. В організації повинні бути особи, які відповідальні за прийняття внесених змін до системи.

7. Відправлення повідомлення обраним користувачам яке матиме інформаційних характер змін файлу в даним момет часу, тобто якщо дані якимось чином були змінени, власник даного файлу отримає нотіфікешн про його зміну з детальною інформацією.

Для організації системи контролю версій треба налаштувати модель де вони будуть зберерігатися, тоба організація взаємовідношин основних обьектів які представляють собою можна сказати реліаційну базу даних в яких я зберігаю стан файлів. Це є необхідною частиною організацію даних для розробки яка буде вулючати в себе всю інформацію про файл, метадату яка була змінена в системі якимось спямованим або ненавмисним чином в системі. Така модель обьектів будет мати наступний вид:

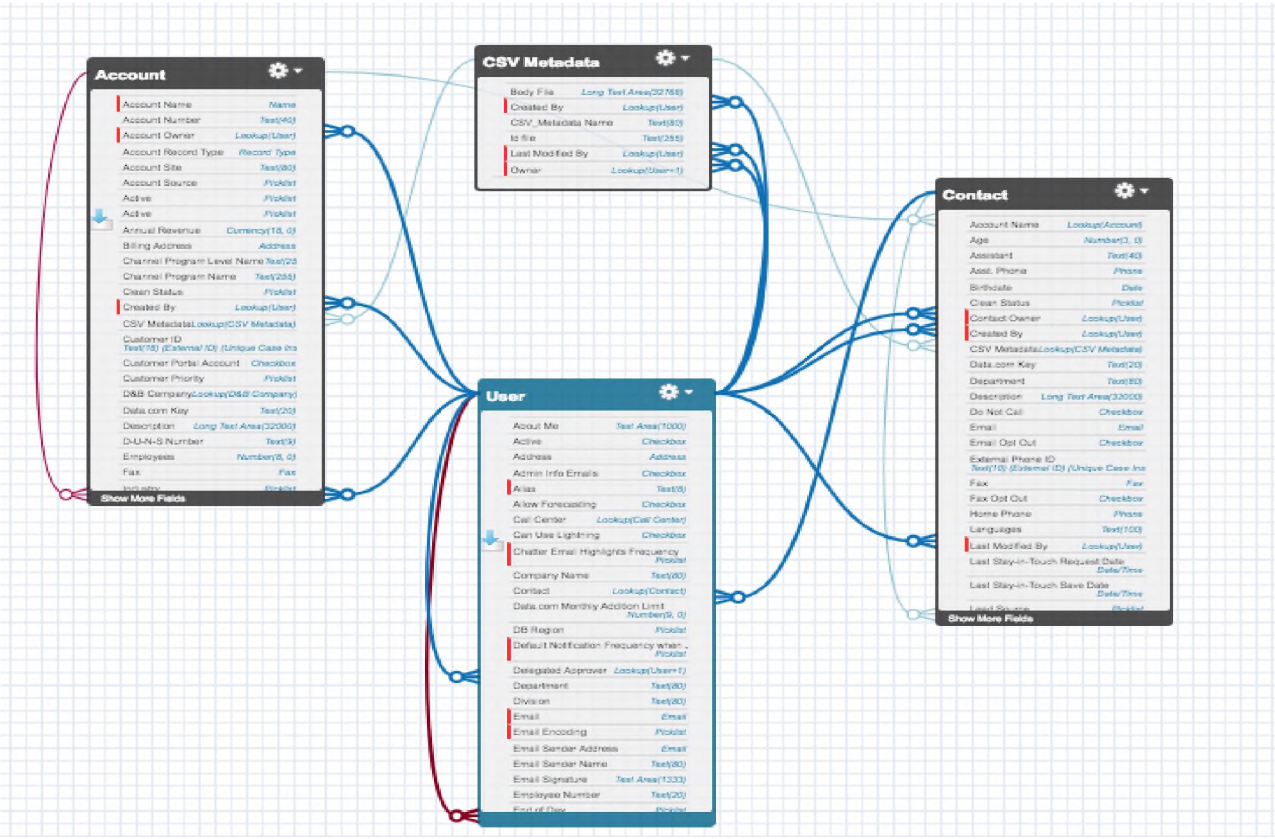

Рисунок — 4.3 Вид таблиці для збереженя контролю версій в SalesForce Системі.

На графіку можна спостерігати відносни таблиць які допомагають мені розуміти ким була зроблена моніпуляція файлу в момент його зміни.

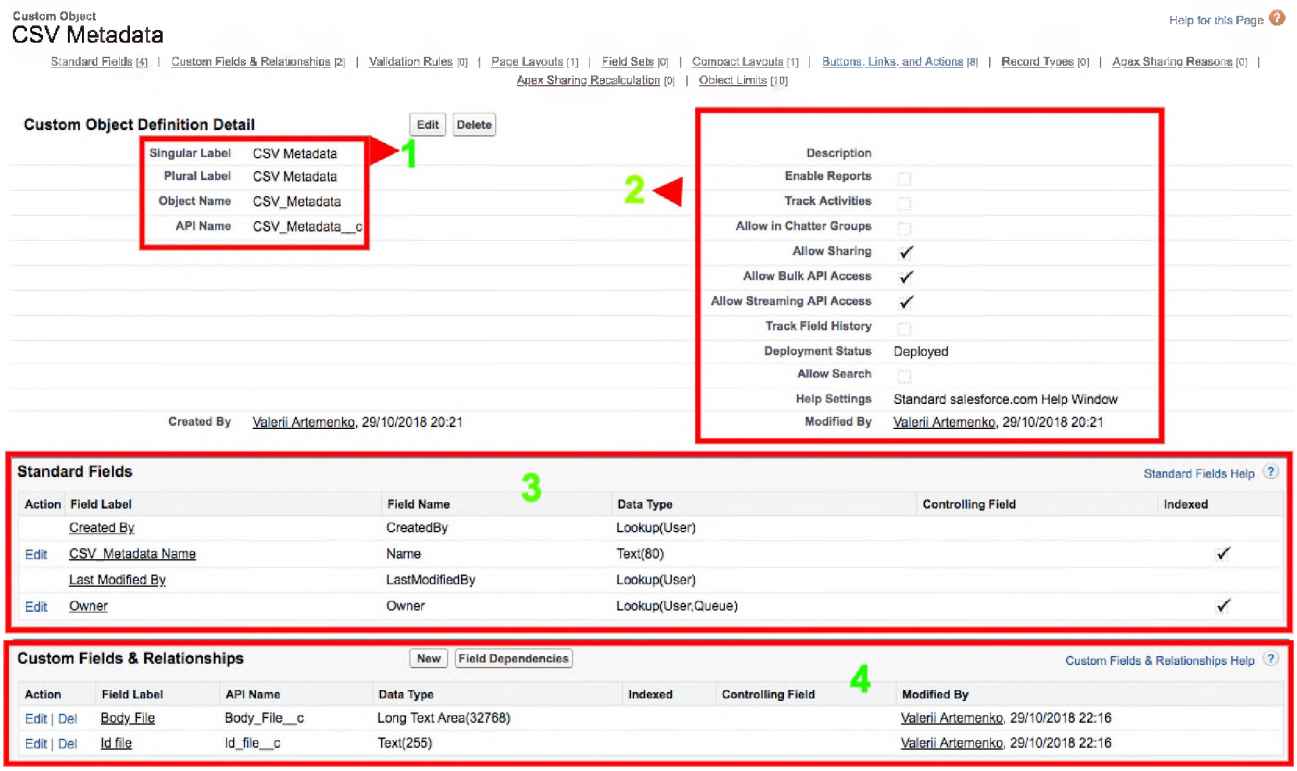

Рисунок — 4.3 Інформація струкрути обьекта збереження контролю версій

На зображені чітко видно структуру обьекта збереження файлів. Метадати або файлів конфігурації. Будь що, що користувач вирішить покласти в даний обьект.

1. Базова інформація нашого обьекту, те що буде батичи користувач системи, це ім’я нашого обькту, в даній розробці вона має назву “CSV Metadata” але назва є неостаточною, лише для демострації наукової роботи.
2. Блок мінімальних налаштувань обьекту які є доступними для будь-якого користувача який буде здатний встановити такі можловості як:
    a. Доступність побудови репорту (що дозволяє будувати шрафічні діаграми з даних обьекту та використовувати в потребах користувача);
    b. Відслідковувати активніть обьекту;
    c. Дозвіл на маштабовану обробку даних, що дозволить обробляти записи купою в одному контексті ;

d. Опис до обьекту що несе інформаційних характер;
e. Дозвіл до обьекту з стороніх систем шляхом інтеграції до стандартної API;
f. Інформація власника обьету;

3. Блок який відповідає за основну поведінку нашого обьекту, тобто місця де зберігаються основна інформація якою виконується поніпуляція версії файлу а саме: тіло файлу, ким було зроблена модифікація, коли була зроблена конкретне творення версії, під яким користувачем була зроблена зміна, GIUD який дає версії унікальність в системі.

Дана реалізація не є межою, що значить наш обьект є маштабованим, де ми можемо зберігати додаткову інформацію яку важатимемо важливою, що є великою перевагою в такій системі в даній розробці.

Рисунок — 4.4 Мокап юзер інтервейсу системи контролю версії файлу

Система має дружелюбну для користування юзер інтерфейс в якому користувач може порівнювати версії файлу з попередніми існуючими версіями в системі і при необхідності повертати стан файлу в вибрану версію. Інструмен

є доволі потужним та простим у використані що є важливим для користувача. В майбутній розробці планужться виконати такі функції як зоиття двух існуючих версій в одну атоматично або вручну.

Висновки до розділу 4

В даному розділі було опрацьована апробація запропанованого архітектурного рішення контролю версій в складі SalesForce системи. Було запропановано алгоритм стиснення даних, який є детально описаним в даному розділі та порівняний з існуючими алгоритмами побудований графік пропускної спрможностістиснення даних та віднощення часу до використанної пам'яті. Побудовані та описані можливості системи конфігурації зробленої розробки. Детально охарактеризована модель даних яка згадувалася в попередньому розділі в якій чітко зображена схема відношень обьектів які беруть учать у робочому процессі. Описана інформаційна структура обьекта який було створено для збереження версій файлу для аудиту. Побудовані блоки які відповідають інформацію обьекта, яка є корисною для користувача,блок мінімальни налаштувань створеного обьекту для котрих зроблино опис та блок в якому описеється основна поведінка нашого обьекту яка є маштабованою дивлячись з потреб користувача. Створена сторінка яка дозволяє користувачу системи порівнювати версіх файлу в зручному для користування графічному інтерфейсі.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

В даному розділі викладено маркетинговий аналіз перспектив реалізації вбудованої системи контролю версій в хмарних системах на прикладі SalesForce CRM системи , а також оцінено можливості її ринкового впровадження.

5.1 Опис ідеї проекту

Проект направлений на забезпечення гнучкого, безпечного і зручного процесу збереження версій файлу в процессі їх обробки з можливістю відновлювати попередні версії файлу. Така система володіє високим рівнем довіри у користувачів, завдяки властивостям які їй надає технологія система контролю верій. В межах її функціоналу є створення офіційних голосувань адміністративним органом та висування другорядних ініціатив будь-яким користувачем системи.

Таблиця 5.1 − Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система контролю версій розробка та впровадження власного інструменту системи керування версіями в SalesForce системі цілю захисту приватних даних та уникнення несовмістності файлів.	1. Проведення процесу збереження стану даних на еожен момент змінення.	Можливість провести конфіденційний та прозорий процес збереження даних між усіма користувачами системи, при цьому кожен користувач зможе зайти і перевірити достовірність результатів голосування

Кінець таблиці 5.1

	2. Висування ініціатив рядовими користувачами	Кожен користувач може виразити свою думку на рахунок будь-яких негативних чи позитивних моментів в організації, висунувши свою власну ініціативу
	3. Кооперація та об’єднання громади	Шляхом збереження версій даних, користувачі зможуть перевіряти стан даних та моніторити історію змін профайлу користувачів.

При порівнянні з конкурентами в першу чергу увага надається архітектурному підходу, що забезпечує більшу захищенність процесу ухвалення рішень і гарантує перевірку достовірності результатів ухвалення при перевірці користувачем. Порівняння з конкуретнами, а також визначення переваг і недоліків наведено у наступні таблиці.

Таблиця 5.2 − Визначення сильних, слабких та нейтральних характеристик ідеї проекту

No п/п	Техніко-економічні характеристики	(потенційні) товари/концепції конку-рентів	W (слабка сторона	N (нейтра-льна	S (силь на

	ідеї	Мій проект	SVN	Mercurial	)	сторона)	сторона)
1	Значне збільшення швидкодії	так	ні	так			так
2	Можливість доступу повна історія розробки доступна оффлайн	так	так	ні			так
3	Розподілена, пірінгова модель	так	ні	так		так	
4	Використання плагінів, а не скриптів	так	ні	ні	так		
5	Менше можливостей для нестандартних рішень	так	ні	ні			
6	Прозорість результатів ухвалення рішення	так	ні	ні			так

7	Розподілена модель системи контролю версій	так	ні	ні			так
8	Можливість швидкого навчання персоналу.	так	ні	ні			так

Конкуренти володіють лише частковим функціоналом, який реалізований в даному проекті. Серед сильних сторін визначені системи контролю версій з доступу повна історія розробки, швидодійність збереження файлів та використання плагінів, а не скриптів що підвищує рівень користування до системи. Слабкою стороною є розподілена модель системи контролю версій, через залежність від платформи SalesForce, на якій реалізована система.

5.2 Технологічний аудит ідеї проекту

Таблиця 5.3 − Технологічна здійсненність ідеї проекту

No п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
	Вбудована система контролю в SalesForce системі	Ідея системи контроля версій	Існуючі платформи, такі як Git, SVN, Mercurial та інші, що дозволяють запровадити свою логіку.	Є доступними та безкоштовними для використання
	Розслідування інцидентів, аудит та консалтинг в сфері інформаційної безпеки	Використання платформи для розслідування інцидентів.	Потрібно придбати ліцензії IBM та телефони	Так, ця технологія є доступною
	Стан фалів.	Централізованийв еб-застосунок	Мови програмування, фреймворки, що дозволяють побудувати веб-системи	Велика кількість фреймворків вільно розповсюджуються і є доступними

	Моніторинг безпеки інформаційних систем клієнтів	Використання програмного забезпечення Splunk Enterprise Security як основного засобу SIEM	Потрібно придбати ліцензію Splunk та налаштувати програмний комплекс	Так, ця технологія є доступною
Обрана технологія реалізації проекту: системи контроля версій				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

При дослідженні ринкових можливостей, в першу чергу проведений аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. Дані наведені у таблиці нижче.

Таблиця 5.4 −Попередня характеристика потенційного ринку стартап-проекту

о п/п	Показники стану ринку (найменування)	Характеристика
	Кількість головних гравців, од	3
	Загальний обсяг продаж	?
	Динаміка ринку (якісна оцінка)	Зростає
	Наявність обмежень для входу	Немає
	Специфічні вимоги для стандартизації, специфікації	Немає
	Середня норма рентабельності в галузі, %	?

Враховуючи сьогоднішню зацікавленість суспільства до хмарних технологій, а також інтерес інвесторів до технології, за попереднім оцінюванням ринок є приваблим для входження, але має ряд обмежень та вимог до сертифікації.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

No п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Аудит версій файлу в змарній системі	• CRM Системи • Приватні підприємства	• Використання системи CRM установами чітко регулюється замовником чи конкретним користувачем системи • Приватні підприємства в свою чергу можуть визначати самостійно формат взаємодії та внести свої коригування	•Впровадження систем контролю версый файли розробки. • Можливість перевірити достовірність процесу збереження контролю версій

Таблиця 5.6 – Фактори загроз

No п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Відсутність зацікавленості у організацій, які не прагнуть до демократизації	Успіх системи багато в чому залежить від підтримки з приватних структур, адже з великою імовірністю організації, які мають в своїй роботі не прозорі процеси, не прагнуть до демократизації	Надання прикладів роботи з її перевагами рядовим працівникам, які в свою чергу будуть спонукати керівництво до впровадження данної системи
2	Новизна технології	Технологія контролю версій, недивлячись на успіх у IT індустрії, ще є молодою і великі гравці ринку лише придивляються до неї. У великій мірі вони ще досліджують можливість реального її застосування і орієнтуються на успішні приклади впровадження, яких ще недостатньо	Розробка широкої інтеграції з існуючими популярними технологіями за для більш плавного впровадження системи

Таблиця 5.7 – Фактори можливостей

No п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Споживча готовність населення	В наш час суспільство стрімко рухається в бік демократизації. Не виключенням є і приватні організації. Прозорий бізнес є фундаментом успіху, тому впровадження систем, які зможуть підняти рівень довіри до організації та внутрішній рівень демократизаціїї в цілому є гарною можливістю вдосконалити свою організацію	Підтримка надійного іміджу компанії, продовження політики інформування населення про можливості впровадженої сстеми контролю версій
2	Зростання популярності технології Blockchain у світі	2002 – теперишниій час ,роки які є піковими популярності технології системи контролю версі, завдяки постійним розробкам. Завдяки цій популярності представники бізнесу	Висвітлення альтернативних сфер використання технології

		активно цікавляться як технологія може удосконалити їх бізнес-процеси та яку вигоду принесе її використання	
3	Прозорість процесу системи контролю версій	Технологія впровадженої системи контролюв ерсій, що лежить в основі стартап-проекту, дозволяє будь-якому користувачу мережі системи SalesForce переглянути стан системи, таким чином досягається прозорість системи, побудованої на основі існючих рішень	Надати користувачам гарантію того, що їх файл точно буде збережено

Кінець таблиці 5.7

Одночасно і можливістю і загрозою є те що технології системи контролю версій є доволю не новою технологією — вона набуває все більшої популярності, але використання її у інших сферах, окрім як для розробки программного забезпечення, все ще не стало достатньо широким.

Таблиця 5.8 − Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції: олігополія	На ринку представлені декілька компаній, що поставляють подібні послуги керування даними	Акцентування переваг продукту, що забезпечує використання технології системи контролю версій
2. Рівень конкурентної боротьби: національний/інтернаціональний	Першим етапом є боротьба за ринок України, де послуги конкурентів не запропоновані з подальшим виходом на ринки інших країн	Маркетингова компанія в першу чергу орієнтована на захоплення місцевого ринку
3. Галузева ознака: внутрішньогалузева	Економічна боротьба з конкурентами відбувається в одній галузі економіки, пропонуються аналогічні послуги, що мають архітектурні відмінності у функціонуванні	Пропозиція суттєвих переваг у порівнянні з продуктами конкурентів у визначеній галузі економіки

Кінець таблиці 5.8

4. Конкуренція за видами товарів: товарно-видова	Конкуренція відбувається між послугами одного виду. За такої конкуренції значення набуває марка товару	Постійна робота над забезпеченням високого рівня іміджу компанії
5. За характером конкурентних переваг: нецінова	Передбачається ведення конкурентної боротьби не за рахунок зниження ціни на аналогічні послуги, а за рахунок новизни та унікальних характеристик технології, на якій базується функціонування системи	Акцент на унікальних характеристиках пропонованого товару
6. За інтенсивністю: марочна	Виведення товару на ринок передбачається під власною маркою, а також створення асоціації між назвою фірми.	Просування продукту компанії під визначеним брендом

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Change.org, Care2	Гнучкі ціни, Патент на продукт	Змінні витрати постачальників	Рівень чутливості до зміни цін	Ціна, лояльність споживачів
Висновки	Конкуренція не інтенсивна, кожен працює в окремому регіоні	Можливість входу в ринок висока. Потенційні конкуренти присутні	Постачальник може диктувати умови: ціни на послуги	Кожен з клієнтів потребує індивідуального підходу для вирішення його задач	Обмежень для роботи на ринку з боку товарів замінників в на даний момент не існує

В результаті проведення аналізу таблиці 5.9, можна зробити висновок, що можливість виходу на ринок з огляду на конкурентну ситуацію є високою. Для виходу на ринок товар в першу чергу повинен пропонувати унікальні характеристики, які відсутні у продуктах конкурентів.

На основі аналізу конкуренції, проведеного в таблиці 5.9, а також із урахуванням характеристик ідеї проекту (таблиця 5.2), вимог споживачів до товару (таблиця 5.5) та факторів маркетингового середовища (таблиці 5.6 та

4.7), визначається та обґрунтовується перелік факторів конкурентоспроможності, що надається у таблиці 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактори конкурентоспроможності	Обґрунтування
1	Динаміка галузі	Технологія система контролю версій наразі є дуже популярною, тому підприємства зацікавлені у ній
2	Концепція товару і послуги	Система керування даними з використанням контролю версій дозволяє досягти прозорості та звільнитися від посередника
3	Післяпродажне обслуговування	Підтримка щодо використання системи після її продажу

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін системи керування приватними даними з використанням технології системи контроля версій.

№ п/п	Фактори конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з власною системою						
			-3	-2	-1	0	1	2	3
1	Динаміка галузі			✓					
2	Концепція товару і послуги				✓				

3	Післяпродажне обслуговування							✓	

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 5.12).

Таблиця 5.12 – SWOT-аналіз стартап проекту

Сильні сторони: - Іноваційні технології - Висока якість	Слабкі сторони: - Слабкий імідж компанії - Слабкий маркетинг - Мало оборотних коштів - Вузька продуктова лінія - Невідома торгівельна марка
Можливості: - Нові технології - Нові потреби клієнтів - Тенденції попиту	Загрози: - Продукти-замінники - Законодавче регулювання - Зміна тенденцій

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент

1	Компанії	Переважно готові	Дуже високий	Низька	Легко

Кінець таблиці 5.13

2	Держустанови	Переважно готові	Дуже високий	Висока	Важко
Цільовими групами обрано компанії, що зацікавлені у демократизації процесу ухвалення рішень, шляхом застосовування автоматизованих систем					

Базові стратегії в обраних сегментах ринку представлені у таблиці 5.14.

Таблиця 5.14 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
1	Динамічний розвиток з використанням маркетингу та встановлення бізнес-контактів	Підняття рейтингу компанії шляхом маркетингу, встановлення конкурентоспроможних цін	Незалежність від посередника, який утримує кошти за свої послуги	Стратегія лідерства по витратах

№ п/п	Обрана альтернатива розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
2	Динамічний розвиток завдяки висвітленню унікальних характеристик надаваємих послуг	Унікальність послуг, що прозорий процес ухвалення рішень	Використання технології блокчейн, що дозволяє зробити процес прозорим та анонімним	Стратегія диференціації

Залежно від міри сформованості галузевого ринку, характеру конкурентної боротьби, необхідно обрати одну з трьох стратегій конкурентної поведінки: розширення первинного попиту, оборонну або наступальну стратегію або ж застосувати демаркетинг або диверсифікацію (таблиця 4.15).

Таблиця 5.15 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект першопрохідцем на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів	Чи буде компанія копіювати основні характеристики	Стратегія конкурентної поведінки

1	Проект не є першопрохідцем	Компанія буде шукати нових користувачів	Компанія буде копіювати найкращі з характеристик конкурентів	Стратегія наслідування лідеру за для економії фінансових ресурсів

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (таблиця 5.5), а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки була розроблена стратегія позиціонування (таблиця 5.16).

Таблиця 5.16 – Визначення стратегії позицінування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Висока доступність та захищенність	Стратегія диференціації	Використання технології блокчейн, що не контролюється жодним органом	Доступність, захищенність, прозорість

5.5 Розроблення маркетингової програми стартап-проекту

Маркетингова програма - це намічений для планомірного здійснення, об'єднаний єдиною метою та залежний від певних строків комплекс взаємопов'язаних завдань і адресних заходів соціального, економічного, науково-технічного, виробничого, організаційного характеру з визначенням ресурсів, що використовуються, а також джерел одержання цих ресурсів. Основну увагу слід приділяти вибору, значенню та формі інструментів маркетингу, їх об'єднанню в найбільш оптимальний з погляду визначеної мети комплекс, а також розподілу фінансових ресурсів у межах бюджетування маркетингу.

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару (таблиця 4.17).

Таблиця 5.17 − Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентом
1	Можливість проведення процесу зьереження версій файлу за будь-яких умов	Доступність, що забезпечується вбудованою системою контролю версій	Невразливість до DoS атак на відміну від централізованих сховищ
2	Чесна та коректна робота системи	Прозорість та можливість перевірити коректність збереження існуючих файлів	Технологія контролю версій дозволяє кожному учаснику системи перевірити історію роботи на файлом(и)
3	Гарантія невтручання	Незалежність системи від посередника	Дані, одного разу записані до контролю версій, ніким не можуть бути змінені чи видалені

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та послуги, його фізичні складові, особливості процесу його надання (таблиця 5.18).

Таблиця 5.18 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Товар забезпечує проведення прозорого та конфіденційного процесу збереження даних.
2. Товар у реальному виконанні	Властивості: доступність, цілісність, конфіденційність, централізованість, прозорість
	Товар представляє собою програмний комплекс з одного модуля: розробленого пакенту для встановлення в систему SalesForce.
	Поставляється у вигляді запакованого веб-застосунку в форматі jar
	Назва: Providing control versions of client data in CRM systems.
3. Товар із підкріпленням	До продажу: відбувається інсталяція та конфігурування системи, проводяться тренінги для клієнта
	Після продажу: відбувається підтримка програмного забезпечення та його допрацювання під потреби клієнта
Система контролю версый, що забезпечує запис даних до системи SalesForce розповсюджується вільно, захисту підлягає модуль веб-системи, кожна копія якого супроводжується вбудованою ліцензією	

Аналіз системи збуту передбачає визначення ефективності кожного елемента цієї системи, оцінювання діяльності апарату працівників збуту. Аналіз витрат обігу передбачає зіставлення фактичних збутових витрат за кожним каналом збуту і видом витрат із запланованими показниками для того, щоб виявити

необґрунтовані витрати, ліквідувати затрати, що виникають у процесі руху товарів і підвищити рентабельність наявної системи збуту.

Дані щодо визначення системи збуту надаються в таблиці 5.19.

Таблиця 5.19 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Власна система збуту	Проведення та розгортання програмного забезпечення на стороні компанії-клієнта	Канал нульового рівня, продажа товару відбувається безпосередньо споживачам через відділ збуту	Оптимальною системою збуту є прямий збут з каналом нульового рівня за відсутності посередників

Таблиця 5.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення
1	Консервативна поведінка, але відкриті до нового,	Соцмережі професійного спрямування, корпоративна пошта	Централізованість, перевіряємість, незалежність від посередника	Висвітлити унікальні характеристики продукту

У якості концепції маркетингових комунікацій були обрані інтегровані маркетингові комунікації, де компанія ретельно обмірковує і координує роботу своїх численних каналів комунікації, рекламу в засобах масової інформації, особистий продаж, стимулювання збуту, пропаганду, прямий маркетинг, упаковку товару.

Висновки до розділу 5

В даному розділі був проведений маркетинговий аналіз перспектив реалізації системи контролю версій для збереження даних в CRM системах на прикладі Salesforce та проведене оцінювання можливостей її ринкового впровадження.

Результатом проведеного аналізу та оцінки ризиків, було виявлено, що даний проект має можливість для ринкової комерціалізації. Для представленого рішення є високий рівень попиту, що пов'язаний із станом інформаційної безпеки користувачів, тому робота над проектом має гарну рентабельність на ринку послуг у сфері кібербезпеки. Для цього стартап-проекту є непогані перспективи входження в ринок, але наявні певні бар'єри, подолання яких підвищую конкурентоспроможність представленого рішення.

Конкурентна ситуація надає перспективи впровадження продукту, так як продукція товарів-аналогів має лише частковий функціонал реалізованої системи та володіє низкою критичних недоліків, через які рівень довіри до них залишається незадовільним. В результаті існуючі товари-аналоги не створюють прямої конкуренції на ринку України. Основною проблемою є можливе негативне ставлення приватних організацій до розвитку в бік демократизації внутрішніх процесів, та до надання рядовим членам організації можливостей впливу на екосистему організації в цілому.

Проведений аналіз підтверджує, що подальша імплементація проекту є доцільною.

ВИСНОВКИ

У діній роботі було досліджено механізм забезпечення цілісності, доступності та спостережності версій клієнтських даних в CRM системах; розроблено рішеня з інтеграції системи керування версіями в CRM систему.

Поставлені та опрацьовані задачи. Визначено основні принципи роботи з клієнтськими даними в CRM системах, вивчення існуючих механізмів забезпечення цілісності, конфіденційності, доступності та спостережності даних в таких системах. Дослідженно принципи роботи систем контролю версій, визначиено види таких систем, основні переваги та недоліки існуючих рішень. Оглянуто основні підходи до інтеграції системи контролю версій в складі CRM системи на прикладі SalesForce;

Були запропоновані елементи системи контролю версій в умовах, типових для даних, що обробляються CRM системою. Зокрема, для вимог цілісності та конфіденційності та доступності версій: обрано метод стиснення даних, шифрування. Запропоновано архітектурні рішення для реалізації контролю версій в складі CRM на прикладі Salsforce системи. Розроблено выдповідний програмний продукт для обраної CRM системи який був успішно імплементований в CRM систему як система котролю версій.

І на закінчення хочеться сказати, що безпека не завжди забезпечується тільки захистом. Вона може бути досягнута також відповідними правилами поведінки і взаємодії об'єктів, високою професійною підготовкою персоналу, безвідмовністю роботи техніки, надійністю всіх видів забезпечення функціонування об'єктів інформаційної безпеки. Конфіденційність інформації - це принцип аудиту, що полягає в тому, що аудитори зобов'язані забезпечувати збереження документів, отриманих або складають ними в ході аудиторської діяльності, і не має права передавати ці документи або їх копії яким би то не було третім особам, або розголошувати усно, що містяться в них відомості без згоди власника економічного суб'єкта. За винятком випадків, передбачених законодавчими актами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Concurrent Versions System. — [Електронний ресурс] — Режим доступу. — URL: http://savannah.nongnu.org/projects/cvs (дата звернення 12.12. 2017)

2. Apache Subversion — [Електронний ресурс] — Режим доступу. — URL: https://subversion.apache.org/ (дата звернення 12.12. 2017)

3. Git — Documentation. — [Електронний ресурс] — Режим доступу. — URL: https://git-scm.com/doc (дата звернення 12.12. 2017)

4. Mercurial — [Електронний ресурс] — Режим доступу. — URL: https://www.mercurial-scm.org/ (дата звернення 12.12. 2017)

5. Google Trends — [Електронний ресурс] — Режим доступу. — URL: https://trends.google.com/trends/explore?date=all&q=%2Fm%2F05vqwg,%2Fm%2F012ct9,%2Fm%2F08441_,%2Fm%2F08w6d6,%2Fm%2F09d6g&hl=en-US&tz=&tz= (дата звернення 25.12.2017)

6. Stack Overflow Survey 2017 — [Електронний ресурс] — Режим доступу. — URL: https://insights.stackoverflow.com/survey/2017 (дата звернення 20.12.2017)

7. Computer Science Center — [Електронний ресурс] — Режим доступу. — URL: https://compscicenter.ru/media/slides/techseminar2014_2014_spring/2014_03_12_techseminar2014_2014_spring.pdf (дата звернення 18.12. 2017)

8. Wikipedia — Comparison of version control software — [Електронний ресурс] — Режим доступу. — URL: https://en.wikipedia.org/wiki/Comparison_of_version_control_software (дата звернення 12.12. 2017)

9. Antamoshkin O., Kukarcev V., Pupkov A., Tsarev R. Intellectual support system of administrative decisions in the big distributed geoinformation systems // International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM 14. - 2014. - С. 227-232. Міжнародний науковий журнал «Синергія наук»

10. Kukartsev V.V. Process programm realization of the capital reproduction funds // Вісник Сибірського державного аерокосмічного університету ім. академіка М.Ф. Решетнева. - 2009. -No 5 (26). - С. 129-132.

11. Єфімов С. Н., Тинченко В. В., Тинченко В. С. Проектування обчислювальної мережі ефективної архітектури для розподіленого вирішення складних завдань // Вісник Сибірського державного аерокосмічного університету ім. академіка МФ Решетнева. - 2007. - No.

12. Тинченко В. С., Тинченко В. В. Особливості застосування GRID-технології для розподіленого рішення великомасштабних завдань // Актуальні проблеми економіки, інформатики та права. - 2008. - С. 132.

13. Шеенок Д.А., Кукарцев В.В. прогнозування вартості розробки систем з програмної надмірністю // Известия Волгоградського державного технічного університету. - 2013. - Т. 17. - No 14 (117). -

14 Литература: 1.Котяшичев И. А. К вопросу о безопасности облачных технологий в информационной среде [Текст] / И. А. Котяшичев, С. В. Смоленцев // Молодой ученый. — 2014. — №5.1. — С. 25-28.

15 Социальные опасности и защита от них: учебник для студ. учреждений высшего проф. образования / [В. М. Губанов, Л. А. Михайлов, В. П. Соломин и др.]; под ред. Л.А. Михайлова. – М., Издательский центр «Академия», 2012.

16. Удо Шнайдер Использование облачных сервисов [Текст] / У. Шнайдер //

17. Журнал сетевых решений/LAN. — 2013. — № 04. 5.Peter Mell, Thimothy Grance. «The NIST Definition of Cloud Computing (Draft)» // Recommendations of the National Institute of Standards and Technology, Special Publication 800 – 145 (Draft), сентябрь 2011 год. Пожалуйста, не забудьте правильно оформить цитату: Котяшичев И. А., Бырылова Е.А.Защита информации в «Облачных

технологиях» как предмет национальной безопасности // Молодой ученый. — 2015. — №6.4. — С. 30-34. — URL https://moluch.ru/archive/86/16357/ (дата обращения: 19.11.2018).

19. Trailhead - Application Lifecycle management module https://trailhead.salesforce.com/en/modules/alm_deployment/units/alm_source_control -

20. Основы Git https://git-scm.com/about

21. Salesforce Development Lifecycle Guide - Chapter 7 https://resources.docs.salesforce.com/sfdc/pdf/salesforce_development_lifecycle.pdf

22. Матеріал з Вікіпедії - вільної енциклопедії - https://ru.wikipedia.org/wiki/Git

23. SalesForce Apex Developer Guide https://developer.salesforce.com/docs/atlas.enus.apexcode.meta/apexcode/apex_dev_guide.htm

24. Gearset whitepaper on release management - Simplifying Salesforce release management: a best-practice approach https://gearset.com/assets/whitepaper-simplifying-salesforce-release-management.pdf

ДОДАТОК А

```
        /**
         * Created by valeriyartemenko on 29.10.18.
         */

        public with sharing class LZString {
            static String keyStr =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=';

            public static String compress(String uncompressed) {

                if (uncompressed == null)
                    return '';

                Integer value;
                Map<String, Integer> context_dictionary = new Map<String,
Integer>();
                Set<String> context_dictionaryToCreate = new Set<String>();

                String context_c = '';
                String context_wc = '';
                String context_w = '';
                double context_enlargeIn = 2; // Compensate for the first entry
which
                // should not count
                Integer context_dictSize = 3;
                Integer context_numBits = 2;
                String context_data_string = '';
                Integer context_data_val = 0;
                Integer context_data_position = 0;


                for (Integer ii = 0; ii < uncompressed.length(); ii += 1) {
                    context_c = String.fromCharArray(new
Integer[]{uncompressed.charAt(ii)});
                    if (!context_dictionary.containsKey(context_c)) {
                        context_dictionary.put(context_c, context_dictSize++);
                        context_dictionaryToCreate.add(context_c);
                    }

                    context_wc = context_w + context_c;

                    if (context_dictionary.containsKey(context_wc)) {
                        context_w = context_wc;
                    } else {
                        if (context_dictionaryToCreate.contains(context_w)) {

                            if (context_w.charAt(0) < 256) {
                                for (Integer i = 0; i < context_numBits; i++) {
                                    context_data_val = (context_data_val << 1);
                                    if (context_data_position == 15) {
                                        context_data_position = 0;
                                        //context_data_string += (char)
context_data_val;
                                        context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
        //                                System.debug('Context data ==>' +
context_data_val);
```

```
        //                                  System.debug('Context ==> ' +
context_data_string);
                                        context_data_val = 0;
                                    } else {
                                        context_data_position++;
                                    }
                                }
                                value = context_w.charAt(0);
                                for (Integer i = 0; i < 8; i++) {
                                    context_data_val = (context_data_val << 1) |
(value & 1);
                                    if (context_data_position == 15) {
                                        context_data_position = 0;
                                        context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
                                        context_data_val = 0;
                                    } else {
                                        context_data_position++;
                                    }
                                    value = value >> 1;
                                }
                            } else {
                                value = 1;
                                for (Integer i = 0; i < context_numBits; i++) {
                                    context_data_val = (context_data_val << 1) |
value;
                                    if (context_data_position == 15) {
                                        context_data_position = 0;
                                        context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});

                                        context_data_val = 0;
                                    } else {
                                        context_data_position++;
                                    }
                                    value = 0;
                                }
                                value = context_w.charAt(0);
                                for (Integer i = 0; i < 16; i++) {
                                    context_data_val = (context_data_val << 1) |
(value & 1);
                                    if (context_data_position == 15) {
                                        context_data_position = 0;
                                        context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
        //                                  System.debug('Context data ==>' +
context_data_val);
        //                                  System.debug('Context ==> ' +
context_data_string);
                                        context_data_val = 0;
                                    } else {
                                        context_data_position++;
                                    }
                                    value = value >> 1;
                                }
                            }
                            context_enlargeIn--;

                            if (Double.valueOf(context_enlargeIn).intValue() ==
0) {
                                context_enlargeIn = Math.pow(2, context_numBits);
                                context_numBits++;
                            }
                            context_dictionaryToCreate.remove(context_w);
```

```
                        } else {
                            value = context_dictionary.get(context_w);
                            for (Integer i = 0; i < context_numBits; i++) {
                                context_data_val = (context_data_val << 1) |
(value & 1);
                                if (context_data_position == 15) {
                                    context_data_position = 0;
                                    //context_data_string += (char)
context_data_val;
                                    context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
       //                               System.debug('Context data ==>' +
context_data_val);
       //                               System.debug('Context ==> ' +
context_data_string);
                                    context_data_val = 0;
                                } else {
                                    context_data_position++;
                                }
                                value = value >> 1;
                            }

                        }
                        context_enlargeIn--;
                        if (Double.valueOf(context_enlargeIn).intValue() == 0) {
                            context_enlargeIn = Math.pow(2, context_numBits);
                            context_numBits++;
                        }
                        // Add wc to the dictionary.
                        context_dictionary.put(context_wc, context_dictSize++);

                        String newstring = context_c;
                        context_w = String.valueOf(newstring);
                    }
                }

                // Output the code for w.
                if (!''.equals(context_w)) {
                    if (context_dictionaryToCreate.contains(context_w)) {
                        if (((Integer) context_w.charAt(0)) < 256) {
                            for (Integer i = 0; i < context_numBits; i++) {
                                context_data_val = (context_data_val << 1);
                                if (context_data_position == 15) {
                                    context_data_position = 0;
                                    //context_data_string += (char)
context_data_val;
                                    context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
       //                               System.debug('Context data ==>' +
context_data_val);
       //                               System.debug('Context ==> ' +
context_data_string);
                                    context_data_val = 0;
                                } else {
                                    context_data_position++;
                                }
                            }
                            value = (Integer) context_w.charAt(0);
                            for (Integer i = 0; i < 8; i++) {
                                context_data_val = (context_data_val << 1) |
(value & 1);
                                if (context_data_position == 15) {
                                    context_data_position = 0;
```

```
                                    //context_data_string += (char)
context_data_val;
                                    context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
       //                             System.debug('Context data ==>' +
context_data_val);
       //                             System.debug('Context ==> ' +
context_data_string);
                                    context_data_val = 0;
                                } else {
                                    context_data_position++;
                                }
                                value = value >> 1;
                            }
                        } else {
                            value = 1;
                            for (Integer i = 0; i < context_numBits; i++) {
                                context_data_val = (context_data_val << 1) |
value;
                                if (context_data_position == 15) {
                                    context_data_position = 0;
                                    //context_data_string += (char)
context_data_val;
                                    context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
       //                             System.debug('Context data ==>' +
context_data_val);
       //                             System.debug('Context ==> ' +
context_data_string);
                                    context_data_val = 0;
                                } else {
                                    context_data_position++;
                                }
                                value = 0;
                            }
                            value = context_w.charAt(0);
                            for (Integer i = 0; i < 16; i++) {
                                context_data_val = (context_data_val << 1) |
(value & 1);
                                if (context_data_position == 15) {
                                    context_data_position = 0;
                                    //context_data_string += (char)
context_data_val;
                                    context_data_string +=
String.fromCharArray(new Integer[]{context_data_val});
       //                             System.debug('Context data ==>' +
context_data_val);
       //                             System.debug('Context ==> ' +
context_data_string);
                                    context_data_val = 0;
                                } else {
                                    context_data_position++;
                                }
                                value = value >> 1;
                            }
                        }
                        context_enlargeIn--;
                        if (Double.valueOf(context_enlargeIn).intValue() == 0) {
                            context_enlargeIn = Math.pow(2, context_numBits);
                            context_numBits++;
                        }
                        context_dictionaryToCreate.remove(context_w);
                    } else {
                        value = context_dictionary.get(context_w);
```

```
                        for (Integer i = 0; i < context_numBits; i++) {
                            context_data_val = (context_data_val << 1) | (value &
1);
                            if (context_data_position == 15) {
                                context_data_position = 0;
                                //context_data_string += (char) context_data_val;
                                context_data_string += String.fromCharArray(new
Integer[]{context_data_val});
       //                           System.debug('Context data ==>' +
context_data_val);
       //                           System.debug('Context ==> ' +
context_data_string);
                                context_data_val = 0;
                            } else {
                                context_data_position++;
                            }
                            value = value >> 1;
                        }

                    }
                    context_enlargeIn--;
                    if (Double.valueOf(context_enlargeIn).intValue() == 0) {
                        context_enlargeIn = Math.pow(2, context_numBits);
                        context_numBits++;
                    }
                }

                // Mark the end of the stream
                value = 2;
                for (Integer i = 0; i < context_numBits; i++) {
                    context_data_val = (context_data_val << 1) | (value & 1);
                    if (context_data_position == 15) {
                        context_data_position = 0;
                        //context_data_string += (char) context_data_val;
                        context_data_string += String.fromCharArray(new
Integer[]{context_data_val});
       //                 System.debug('Context data ==>' + context_data_val);
       //                 System.debug('Context ==> ' + context_data_string);
                        context_data_val = 0;
                    } else {
                        context_data_position++;
                    }
                    value = value >> 1;
                }

                // Flush the last char
                while (true) {
                    context_data_val = (context_data_val << 1);
                    if (context_data_position == 15) {
                        //context_data_string += (char) context_data_val;
                        context_data_string += String.fromCharArray(new
Integer[]{context_data_val});
       //                 System.debug('Context data ==>' + context_data_val);
       //                 System.debug('Context ==> ' + context_data_string);
                        break;
                    } else
                            context_data_position++;
                }
                return context_data_string;
            }

            public static String decompress(String compressed){
                if (compressed == null)
                    return '';
```

```
                if (compressed == '')
                    return null;

        //        System.debug('Input Compressed ==>' + compressed);

                List<String> dictionary = new String[200];
        //        for(integer j = 0; j < dictionary.size(); j++)
        //            dictionary.set(j, '');

                double enlargeIn = 4;
                Integer dictSize = 4;
                Integer numBits = 3;
                Integer next;
                String entry = '';

                String result = '';

                String w;
                Integer bits;
                Integer resb;
                double maxpower;
                Integer power;
                String c = '';
                Integer d;

                Data data = LZString.getNewDataInsance();

                data.dataString = compressed;
                data.val = (Integer) compressed.charAt(0);
                data.position = 32768;
                data.index = 1;

                for (Integer i = 0; i < 3; i += 1) {
                    dictionary.add(i, String.valueOf(i));
                }

                bits = 0;
                maxpower = Math.pow(2, 2);
                power = 1;

                while (power != Double.valueOf(maxpower).intValue()) {
                    resb = data.val & data.position;
                    data.position >>= 1;
                    if (data.position == 0) {
                        data.position = 32768;
                        data.val = (Integer)
data.dataString.charAt(data.index++);
                    }
                    bits |= (resb > 0 ? 1 : 0) * power;
                    power <<= 1;
                }

                System.debug('Bits ==>' + bits);
                next = bits;
                if(next == 0){
                    bits = 0;
                    maxpower = Math.pow(2, 8);
                    power = 1;
                    while (power != Double.valueOf(maxpower).intValue()) {
                        resb = data.val & data.position;
                        data.position >>= 1;
                        if (data.position == 0) {
                            data.position = 32768;
```

```
                            data.val = (Integer)
data.dataString.charAt(data.index++);
                        }
                        bits |= (resb > 0 ? 1 : 0) * power;
                        power <<= 1;
                    }
                    //c += (char) bits;
                    c += String.fromCharArray(new Integer[]{bits});
                }
                else if(next == 1){
                    bits = 0;
                    maxpower = Math.pow(2, 16);
                    power = 1;
                    while (power != Double.valueOf(maxpower).intValue()) {
                        resb = data.val & data.position;
                        data.position >>= 1;
                        if (data.position == 0) {
                            data.position = 32768;
                            data.val = (Integer)
data.dataString.charAt(data.index++);
                        }
                        bits |= (resb > 0 ? 1 : 0) * power;
                        power <<= 1;
                    }
                    //c += (char) bits;

                    c += String.fromCharArray(new Integer[]{bits});
                }
                else if(next == 2)
                    return '';

                dictionary.add(3, c);
        //        System.debug('Dictionary before while-true ==>' + dictionary);
                w = c;
                //result = new StringBuilder(200);
                //result = '';

                //result.append(c);
                result += c;
        //        System.debug('result before while-true ==>' + result);
        //        System.debug('Data ==>' + data);

                // w = result = c;

                while(true){
                    if (data.index > data.dataString.length()) {
                        return '';
                    }

                    bits = 0;
                    maxpower = Math.pow(2, numBits);
                    power = 1;
                    while (power != Double.valueOf(maxpower).intValue()) {
                        resb = data.val & data.position;
                        data.position >>= 1;
                        if (data.position == 0 && data.index <
data.dataString.length()) {
                            data.position = 32768;
                            data.val = (Integer)
data.dataString.charAt(data.index++);
                        }
                        bits |= (resb > 0 ? 1 : 0) * power;
                        power <<= 1;
                    }
```

```
        //              System.debug('Data Position 0 ==>' + data.position);
        //              System.debug('Bits 0 ==>' + bits);
                    d = bits;

                    if(d == 0){
                        bits = 0;
                        maxpower = Math.pow(2, 8);
                        power = 1;

        //                System.debug('Data 0 ==> String='+
EncodingUtil.base64Encode(Blob.valueOf(data.dataString)) +
        //                ',position='+data.position+
        //                ',val='+data.val+
        //                ',index='+data.index
        //                );

                        while (power != Double.valueOf(maxpower).intValue()) {
                            resb = data.val & data.position;
                            data.position >>= 1;
                            if (data.position == 0) {
                                data.position = 32768;
                                data.val = (Integer)
data.dataString.charAt(data.index++);
                            }
                            bits |= (resb > 0 ? 1 : 0) * power;
                            power <<= 1;
                        }

                        //temp += (char) bits;
                        String temp = String.fromCharArray(new Integer[]{bits});

        //                if(dictSize > 15)
        //                    System.debug('d-15:' + dictionary.get(15));
                        dictionary.add(dictSize++, String.fromCharArray(new
Integer[]{bits}));

        //                System.debug('Data Position 1 ==>' + data.position);
        //                System.debug('Bits 1 ==>' + bits);
        //                System.debug('Temp 1 ==>' + temp);

                        d = dictSize - 1;

                        enlargeIn--;
                    }
                    else if(d == 1){
                        bits = 0;
                        maxpower = Math.pow(2, 16);
                        power = 1;
                        while (power != Double.valueOf(maxpower).intValue()) {
                            resb = data.val & data.position;
                            data.position >>= 1;
                            if (data.position == 0) {
                                data.position = 32768;
                                data.val = (Integer)
data.dataString.charAt(data.index++);
                            }
                            bits |= (resb > 0 ? 1 : 0) * power;
                            power <<= 1;
                        }

                        //temp='';
                        //temp += (char) bits;
                        String temp = String.fromCharArray(new Integer[]{bits});
```

```
        //                if(dictSize > 15)
        //                    System.debug('d-15:' + dictionary.get(15));
                        dictionary.add(dictSize++, String.fromCharArray(new
Integer[]{bits}));

        //                System.debug('Data Position 2 ==>' + data.position);
        //                System.debug('Bits 2 ==>' + bits);
        //                System.debug('Temp 2 ==>' + temp);

                        d = dictSize - 1;
                        enlargeIn--;
                    }
                    else if(d == 2){
                        return String.valueOf(result);
                    }


                    if (Double.valueOf(enlargeIn).intValue() == 0) {
                        enlargeIn = Math.pow(2, numBits);
                        numBits++;
                    }

                    if (d < dictionary.size() && dictionary.get(d) != null) {
                        entry = dictionary.get(d);
        //                System.debug('d-index ==>' + d);
        //                System.debug('Entry 0 ==>' + entry);
                    } else {
                        if (d == dictSize) {
                            //entry = w + w.charAt(0);
                            entry = w + String.fromCharArray(new
Integer[]{w.charAt(0)});
        //                    System.debug('Entry 1 ==>' + entry);
                        } else {
        //                    System.debug('Entry Failed d ==>' + d);
        //                    System.debug('Entry Failed dictSize ==>' +
dictSize);
                            return null;
                        }
                    }


        //            System.debug('Entry 2 ==>' + entry);
                    result += entry;

                    // Add w+entry[0] to the dictionary.

                    //integer new_W = w.charAt(0) + entry.charAt(0);
                    //dictionary.add(dictSize++, w + String.fromCharArray(new
Integer[]{entry.charAt(0)}));
                    dictionary.add(dictSize++, w + String.fromCharArray(new
Integer[]{entry.charAt(0)}));
                    enlargeIn--;

                    w = entry;

                    if (Double.valueOf(enlargeIn).intValue() == 0) {
                        enlargeIn = Math.pow(2, numBits);
                        numBits++;
                    }
                }

                return result;
            }
```

```
            public static String decompressHexString(String hexString) {

                if (hexString == null) {
                    return '';
                }

                if (Math.mod(hexString.length(), 2) != 0) {
                    //throw new RuntimeException('Input string length should be
divisible by two');
                }

                Integer []intArr = new Integer[hexString.length() / 2];

                Integer i = 0, k = 0;
                for (i = 0; i < hexString.length(); i += 2) {
                    //intArr[k] = Integer.parseInt('' + hexString.charAt(i) +
hexString.charAt(i + 1), 16);
                    String hexaInt = EncodingUtil.convertToHex(Blob.valueOf('' +
hexString.charAt(i) + hexString.charAt(i + 1)));
                    intArr[k] = Integer.valueOf(hexaInt);
                    k++;
                }

                String sb = '';
                for (Integer j = 0; j < intArr.size(); j += 2) {
                    //sb.append(Character.toChars(intArr[j] | intArr[j + 1] <<
8));
                    integer intVal = intArr[j] | intArr[j + 1] << 8;
                    sb += String.fromCharArray(new Integer[]{
                            intVal
                    });
                }


                return decompress(sb) ;
            }

            public static String compressToUTF16(String input) {
                if (input == null)
                    return '';
                String output = '';
                integer c;
                integer current = 0;
                integer status = 0;


                input = LZString.compress(input);

                for (integer i = 0; i < input.length(); i++) {
                    c = (integer) input.charAt(i);
                    status++;

                    if(status == 0) {
                        output += getASCIIValue(((c >> 1) + 32));
                        current = (c & 1) << 14;
                    }
                    else if(status == 1) {
                        output += getASCIIValue(((current + (c >> 2)) + 32));
                        current = (c & 3) << 13;
                    }
                    else if(status == 2) {
                        output += getASCIIValue(((current + (c >> 3)) + 32));
                        current = (c & 7) << 12;
                    }
```

```
            else if(status == 3) {
                output += getASCIIValue(((current + (c >> 4)) + 32));
                current = (c & 15) << 11;
            }
            else if(status == 4) {
                output += getASCIIValue(((current + (c >> 5)) + 32));
                current = (c & 31) << 10;
            }
            else if(status == 5) {
                output += getASCIIValue(((current + (c >> 6)) + 32));
                current = (c & 63) << 9;
            }
            else if(status == 6) {
                output += getASCIIValue(((current + (c >> 7)) + 32));
                current = (c & 127) << 8;
            }
            else if(status == 7) {
                output += getASCIIValue(((current + (c >> 8)) + 32));
                current = (c & 255) << 7;
            }
            else if(status == 8) {
                output += getASCIIValue(((current + (c >> 9)) + 32));
                current = (c & 511) << 6;
            }
            else if(status == 9) {
                output += getASCIIValue(((current + (c >> 10)) + 32));
                current = (c & 1023) << 5;
            }
            else if(status == 10) {
                output += getASCIIValue(((current + (c >> 11)) + 32));
                current = (c & 2047) << 4;
            }
            else if(status == 11) {
                output += getASCIIValue(((current + (c >> 12)) + 32));
                current = (c & 4095) << 3;
            }
            else if(status == 12) {
                output += getASCIIValue(((current + (c >> 13)) + 32));
                current = (c & 8191) << 2;
            }
            else if(status == 13) {
                output += getASCIIValue(((current + (c >> 14)) + 32));
                current = (c & 16383) << 1;
            }
            else if(status == 14) {
                output += getASCIIValue(((current + (c >> 15)) + 32));
                output += getASCIIValue(((c & 32767) + 32));

                status = 0;
            }else {

            }
        }

        output += getASCIIValue(current + 32);

        return output;
    }

    public static String decompressFromUTF16(String input) {
        if (input == null)
            return '';

        String output = '';
```

```
    Integer current = 0, c, status = 0, i = 0;
    while (i < input.length()) {
        c = (((Integer) input.charAt(i)) - 32);

        status++;
        if (status == 0) {
            current = c << 1;
        } else if (status == 1) {
            output += getASCIIValue((current | (c >> 14)));
            current = (c & 16383) << 2;
        } else if (status == 2) {
            output += getASCIIValue((current | (c >> 13)));
            current = (c & 8191) << 3;
        } else if (status == 3) {
            output += getASCIIValue((current | (c >> 12)));
            current = (c & 4095) << 4;
        } else if (status == 4) {
            output += getASCIIValue((current | (c >> 11)));
            current = (c & 2047) << 5;
        } else if (status == 5) {
            output += getASCIIValue((current | (c >> 10)));
            current = (c & 1023) << 6;
        } else if (status == 6) {
            output += getASCIIValue((current | (c >> 9)));
            current = (c & 511) << 7;
        } else if (status == 7) {
            output += getASCIIValue((current | (c >> 8)));
            current = (c & 255) << 8;
        } else if (status == 8) {
            output += getASCIIValue((current | (c >> 7)));
            current = (c & 127) << 9;
        } else if (status == 9) {
            output += getASCIIValue((current | (c >> 6)));
            current = (c & 63) << 10;
        } else if (status == 10) {
            output += getASCIIValue((current | (c >> 5)));
            current = (c & 31) << 11;
        } else if (status == 11) {
            output += getASCIIValue((current | (c >> 4)));
            current = (c & 15) << 12;
        } else if (status == 12) {
            output += getASCIIValue((current | (c >> 3)));
            current = (c & 7) << 13;
        } else if (status == 13) {
            output += getASCIIValue((current | (c >> 2)));
            current = (c & 3) << 14;
        } else if (status == 14) {
            output += getASCIIValue((current | (c >> 1)));
            current = (c & 1) << 15;
        } else if (status == 15) {
            output += getASCIIValue((current | c));
            status = 0;
        }

        i++;
    }

    System.debug('UTF16' + output);
    return LZString.decompress(output);
}

private static String getASCIIValue(integer cInteger){
    return String.fromCharArray(new Integer[]{cInteger});
}
```

```
            public static String compressToBase64(String input) {
                return encode64(compress(input));
            }

            public static String decompressFromBase64(String input) {
                return LZString.decompress(decode64(input));
            }

            public static String decode64(String input) {

                //return
String.valueOf(EncodingUtil.base64Decode(EncodingUtil.urlEncode(input, 'utf-
8')).toString());
                String str = '';

                integer ol = 0;
                integer output_1=0;
                integer chr1, chr2, chr3;
                integer enc1, enc2, enc3, enc4;
                integer i = 0;
                integer j=0;

                while (i < input.length()) {

                    enc1 = keyStr.indexOf(String.fromCharArray(new
Integer[]{input.charAt(i++)}));
                    enc2 = keyStr.indexOf(String.fromCharArray(new
Integer[]{input.charAt(i++)}));
                    enc3 = keyStr.indexOf(String.fromCharArray(new
Integer[]{input.charAt(i++)}));
                    enc4 = keyStr.indexOf(String.fromCharArray(new
Integer[]{input.charAt(i++)}));

                    chr1 = (enc1 << 2) | (enc2 >> 4);
                    chr2 = ((enc2 & 15) << 4) | (enc3 >> 2);
                    chr3 = ((enc3 & 3) << 6) | enc4;

                    if (Math.mod(ol, 2)==0) {
                        output_1 = chr1 << 8;

                        if (enc3 != 64) {
                            //str.append((char) (output_1 | chr2));
                            str += String.fromCharArray(new Integer[]{output_1 |
chr2});
                        }
                        if (enc4 != 64) {
                            output_1 = chr3 << 8;
                        }
                    } else {
                        //str.append((char) (output_1 | chr1));
                        str +=  String.fromCharArray(new Integer[]{output_1 |
chr1});

                        if (enc3 != 64) {
                            output_1 = chr2 << 8;
                        }
                        if (enc4 != 64) {
                            str += String.fromCharArray(new Integer[]{output_1 |
chr3});
                        }
                    }
                    ol+=3;
                }
```

```
        return str;
    }

    public static String encode64(String input) {
        return EncodingUtil.base64Encode(Blob.valueOf(input));
    }


    class Data {
        public integer val;
        public String dataString;
        public integer position;
        public integer index;
    }
    private static Data getNewDataInsance(){
        return new Data();
    }

}
```